Challenges & Opportunities Concerning Corporate Formation, Nonprofit Status, & Governance for Open Source Projects

Organization & Structure of Open Source Software Development Initiatives

Challenges & Opportunities Concerning Corporate Formation, Nonprofit Status, & Governance for Open Source Projects

Dalia Topelson Ritvo
Kira Hessekiel
Christopher T. Bavitz

Harvard Law School
Berkman Klein Center for Internet & Society

March 2017

CYBERLAW CLINIC
HARVARD LAW SCHOOL | BERKMAN KLEIN CENTER FOR INTERNET & SOCIETY

Table of Contents

3 OVERVIEW & BACKGROUND ACKNOWLEDGMENTS

4 PART I: THE QUESTION OF INCORPORATION FOR OPEN SOURCE SOFTWARE INITIATIVES
- Introduction 4
- 501(c)3 Status, the IRS, and Open Source 4
- Shift in IRS Policy 5
- Conclusion 5

6 BEYOND 501(C)(3): STRUCTURAL CONSIDERATIONS FOR OPEN SOURCE SOFTWARE INITIATIVES
- Introduction 6
- Alternative Federal Tax-Exempt Status Recognitions: 501(c)(4) & 501(c)(6) 6
- Nonprofit Corporations 7
- For-Profit Entities: Corporations and Limited Liability Companies 7
- A Third Way: Benefit Corporations 8
- Conclusion 9

10 PART II: GOVERNANCE MODELS FROM THE OPEN SOURCE WORLD & BEYOND
- Introduction 10
- Levels of Control & Openness 10
- Model A: Benevolent Dictatorship 11
- Model B: Meritocracy 13
- Model C: Delegated Governance 15
- Model D: Dynamic Governance 17
- All About Boards: Examples from the Nonprofit World 19
- Norms and Attitudes for a Successful Open Source Software Organization 20

22 CONCLUSION

23 ENDNOTES

27 CASE STUDIES & GOVERNANCE MODEL ILLUSTRATIONS

Suggested Citation: Bavitz, Christopher; Topelson Ritvo, Dalia & Hessekiel, Kira. Organization & Structure of Open Source Software Development Initiatives: Challenges & Opportunities Concerning Corporate Formation, Nonprofit Status, & Governance for Open Source Projects (March 2017). Berkman Klein Center Research Publication. Available at: https://dash.harvard.edu/handle/1/30805146

Cyberlaw Clinic
1585 Massachusetts Avenue | Suite 5018 | Cambridge, Massachusetts 02138
+1 617.384.9125 | +1 617.495.7641 (fax)
https://clinic.cyber.harvard.edu
clinic@cyber.harvard.edu

OVERVIEW & BACKGROUND

This report addresses a number of key considerations that those managing open source software development initiatives should take into account when thinking about structure, organization, and governance. The genesis of this project involved an investigation into anecdotal reports that companies and other institutions developing open source software were facing difficulties obtaining tax exempt nonprofit status under Section 501(c)(3) of Title 26 of the United States Code. Based on conversations with a number of constituents in the open source software development community, the authors have prepared this report to address specific questions about nonprofit status alongside questions about corporate formation and governance models more generally.

Nothing in this report should be viewed as a substitute for specific legal advice on the narrow questions facing particular organizations under particular sets of factual circumstances. But, the authors are hopeful the document provides a general overview of the complex issues that open source initiatives face when balancing a need for structure and continuity with the innovative and experimental spirit at the heart of many open source development projects.

The report has two primary parts:

- First, it addresses some formal organizational considerations that open source software initiatives should weigh, evaluating the benefits of taking on a formal structure and the options for doing so. The report provides information about different types of corporate organization that open source projects may wish to consider. And, it delves into Internal Revenue Service policy and practice and US tax law concerning questions about the tax exemptions referenced above.

- In its second half, the authors pull back to consider more broadly questions of organizational structure, offering ideas about governance models that open source organizations may wish to explore, separate from formal corporate structure, as they seek to achieve their missions.

Different considerations may inform the choice of formal, legal organizational structures (on the one hand) and governance models (on the other hand). By addressing both, the authors hope that this report will be useful to the broadest possible range of managers of and contributors to open source development initiatives.

ACKNOWLEDGMENTS

This report was produced with generous support from the MacArthur Foundation, which allowed the Cyberlaw Clinic at the Berkman Klein Center for Internet & Society to host a small gathering in April 2016 during which stakeholders surfaced questions and concerns they have faced when operating in this space. The authors would like to thank MacArthur and the participants in that gathering for their contributions to the conversation and, in particular, express their appreciation for the input of Brian Behlendorf; Nathan Freitas; Esther Lim; Ellen Lubell; Allison Randal; Wendy Seltzer; Tom Stites; and Aaron Williamson. Nadia Eghbal and Jason Griffey offered valuable contributions as well. Finally, Harvard Law School students enrolled in the Cyberlaw Clinic made significant contributions to this report, including Jin-Kyu Baek; Sarah Baugh; Daniele Kleiner Fontes; Marco Medellin; and Joe Milner.

PART I

THE QUESTION OF INCORPORATION FOR OPEN SOURCE SOFTWARE INITIATIVES

Introduction

Corporate formation may not be the first item on the to-do list of an open source project just beginning to expand. But, having a formal business structure can confer benefits that open source projects might find desirable as they plan their growth.

First, a corporate entity can shield the organization's founders and members from personal liability should someone bring a suit against the project. The existence of a legal entity also helps delineate ownership of the project's assets, which can help open source projects sort questions of intellectual property more clearly. In addition, all corporate formation options available in the United States come with stipulations on how entities should be run and managed. These requirements create structural support, governance and process rules, and checks and balances for how an entity should operate and make decisions. Those tools, in turn, can help promote the longevity of an open source project and protect it from organizational turmoil regarding decision-making authority, ownership of ideas, and financial questions.

Many open source initiatives have a desire to adopt a mission-driven business model because they intend for their work to give back to the open source development community and serve the public good. Following the example of some of the most established open source organizations, many attempt to form as a nonprofit corporations with tax-exempt status. While there are benefits to achieving so-called 501(c)(3) status, several alternative models of corporate formation exist that open source projects would be wise to consider because they may align better with a particular project's goals.

In this section, the authors address several varieties of nonprofit designation (including tax-exempt status), for-profit corporation models, and hybrid benefit corporations to help open source projects understand how to best utilize the advantages of corporate formation to suit their specific needs.

501(c)3 Status, the IRS, and Open Source

Open source projects often seek to follow in the footsteps of well-known organizations such as Apache and Mozilla and seek nonprofit status. This approach may be driven by a sense that nonprofit status exempts an organization from paying federal and state income taxes. But, that is not necessarily the case. In fact, an organization must qualify for a tax exemption under federal tax law.

The Internal Revenue Code exempts *certain* nonprofit organizations, including "[c]orporations, and any community chest, fund, or foundation, organized and operated exclusively for religious, charitable, scientific, testing for public safety, literary, or educational purposes," from paying federal income taxes.[1] Because this specific exemption appears in 26 U.S.C. 501(c)(3), people often refer to organizations that receive the exemption as "501(c)(3)s" or as having "501(c)(3) status."

Historically, some open source organizations have preferred the 501(c)(3) exemption to other 501(c) alternatives.[2] Nonprofit open source organizations generally provide their software to the public for free.[3] These organizations may rely on donations in order to pay for any organizational expenses.[4] Donors prefer to make donations when they can take deductions for their donations and as a result are more likely to make donations to nonprofits with 501(c)(3) status.[5] As a result, in order to make their organizations attractive to potential donors, nonprofit open source organizations often have sought the 501(c)(3) exemption.

At the beginning of the open source software movement, open source organizations attained the 501(c)(3) exemption without much difficulty. Open source organizations began to apply for 501(c)(3) status as early as 2001.[6] The Mozilla Foundation received an Internal Revenue Service ("IRS") determination of its 501(c)(3) status within six months of submitting its application.[7]

Over the course of the 2000s, open source organizations were able to obtain 501(c)(3) status on a regular basis. A 2008 report by the Software Freedom Law Center encouraged open source organization founders to consider applying for 501(c)(3) status on the basis that "[t]he IRS has regularly recognized organizations created for the promotion of free and open source software projects as having charitable purposes."[8] The accompanying Case Study #1, which examines the Apache Software Foundation, illustrates the ease with which open source software organizations were able to obtain 501(c)(3) status during this period.

Shift in IRS Policy

In the past few years, open source software organizations have begun to experience greater difficulty in obtaining 501(c)(3) status. For example, the Open Source Elections Technology Foundation (OSET) (formerly the Open Source Digital Voting Foundation) applied for 501(c)(3) status in 2007.[9] Although OSET eventually received 501(c)(3) status, the IRS review of OSET's application lasted for over six years, and OSET's application was initially denied.[10] OSET's experience is not exceptional. According to Wired, "numerous open source projects" reported that the IRS had "put a freeze on their nonprofit applications."[11]

The change in treatment of open source software companies arose directly from a shift in IRS policy. During the scandal surrounding IRS denial of 501(c)(3) status to Tea Party organizations, the United States House of Representatives Ways and Means Committee released the IRS's "Be on the Look Out" lists ("BOLOs").[12] The BOLOs listed kinds of organizations that the IRS had identified as potentially unsuitable for 501(c)(3) status.[13] Every released BOLO dated between August 2010 and July 2012 included open source organizations.[14]

The lists explain that open source software organizations should be monitored because "[t]he members of these organizations are usually the for-profit business or for-profit support technicians of the software. The software is provided for free, however; fees are charged for technical support by the for-profit."[15] The BOLOs thus misrepresented the wide variety of open source software business models, ignoring that many open source software organizations do not have affiliations with for-profit businesses. Nevertheless, the BOLOs drove IRS policy for at least two years.

While the IRS has discontinued its use of BOLOs,[16] the shift toward more stringent reviews of 501(c)(3) applications remains. OSET was able to obtain 501(c)(3) status, but many open source software organizations still are waiting for determinations.[17]

The accompanying Case Study #2, Case Study #3, and Case Study #4 examine open source software organizations' applications for 501(c)(3) status filed in the past five years. Together, these examples illustrate the trend toward more restrictive IRS review of such applications, underscore the fact that an organization's mission remains key in the context of such review by the Internal Revenue Service, and drive home the point that such mission must go beyond mere provision of free code by promoting a greater educational or social good.

Conclusion

Since the beginning of the 21st century, open source software organizations have pursued 501(c)(3) status in order to operate on nonprofit, donations-based business models. 501(c)(3) status exempted these organizations from federal income tax and made donations to the organizations tax-deductible for the donors. Originally, open source stalwarts like Apache obtained 501(c)(3) status with ease. However, the past few years have witnessed a sea change in IRS treatment of nonprofit open source software organizations. Based allegedly on suspicions of abuse of 501(c)(3) status by nonprofit open source software organizations affiliated with for-profit software developers, the IRS has raised its threshold for granting such nonprofits 501(c)(3) status.

In the wake of recent delays and rejections of 501(c)(3) applications, 501(c)(3) status appears harder to obtain than ever for open source software organizations. While some open source software organizations may continue to obtain 501(c)(3) status, this option may be closed off – in particular, for organizations that choose not to or cannot offer activities to educate the public in software development or a demonstrable

connection to a charitable purpose. Given these changes, open source software organizations can no longer count on obtaining 501(c)(3) status and may want to evaluate the pros and cons of adopting alternative business structures.

BEYOND 501(C)(3): STRUCTURAL CONSIDERATIONS FOR OPEN SOURCE SOFTWARE INITIATIVES

Introduction

As open source software organizations face greater difficulty in obtaining 501(c)(3) status, some organizations may begin to consider alternative options to the traditional 501(c)(3) nonprofit model. A broad spectrum of options exists; some options would require only a small shift in operations, while others would force organizations to embrace new models altogether. This section outlines some of these entity formation options available to open source software initiatives and considers how those managing open source projects might use these options to achieve their goals.

Alternative Federal Tax-Exempt Status Recognitions: 501(c)(4) & 501(c)(6)

Certain open source software organizations may be able to find limited opportunities for federal tax-exempt status by seeking recognition under other subprovisions of Section 501. With 501(c)(3) status increasingly unattainable, open source software organizations that wish to obtain federal tax exempt status may need to consider federal tax exemption under 501(c)(4) and 501(c)(6).[18]

Like 501(c)(3), organizations recognized under 501(c)(4) and 501(c)(6) are exempt from federal income tax.[19] However, donations to organizations recognized under 501(c)(4) and 501(c)(6) are not tax deductible.[20] As discussed above, tax deductibility incentivizes donors to donate to open source software organizations. Without this benefit, operating on a donations-based nonprofit business model may become more difficult.

Moreover, recognition under 501(c)(4) and 501(c)(6) is available only to certain kinds of organizations. 501(c)(4) status is reserved for "civic leagues or organizations . . . devoted exclusively to charitable, educational, or recreational purposes."[21] As discussed above, the IRS has become reluctant to find that open source software organizations applying for 501(c)(3) status operate for charitable or educational purposes. Furthermore, IRS guidance states that an organization created primarily for recreational purposes should apply for exemption under 501(c)(7) rather than 501(c)(4).[22] As a result, most open source software organizations probably should not rely on 501(c)(4) for obtaining tax-exempt status.

Some types of open source software organizations have had some success obtaining federal income tax exemptions under 501(c)(6). 501(c)(6) exempts business leagues from federal income tax.[23] IRS guidance explains that people found business leagues in order to promote a business interest that the founders share in common.[24] In the open source software community, 501(c)(6) organizations generally act as umbrella organizations for many projects rather than working directly on a single project. For example, the Dojo Foundation and the Eclipse Foundation are 501(c)(6) organizations that support open source software projects.[25] But, a shift in IRS policy toward open source software organizations applying for 501(c)(6) status – which parallels the shift regarding 501(c)(3) applications – may be underway.

The OpenStack Foundation, which supports the OpenStack ecosystem, was recently denied 501(c)(6) status. According to Mark McLoughlin, an OpenStack Foundation board member, the IRS argued that the foundation is engaging in business regularly carried on for profit and is improving business conditions only for its members rather than for the entire industry.[26] Jonathon Bryce, the executive director of the Foundation, noted that the Foundation is "fundamentally no different from other similar [open source software] 501(c)(6) organizations."[27] While Bryce offered this statement to support the argument that the OpenStack Foundation eventually will receive 501(c)(6) status, the same point could favor a shift in IRS policy toward refusing to grant 501(c)(6) status and revoking 501(c)(6)

status from other open source software umbrella organizations.

Given the complexities associated with achieving exemptions from federal income tax, open source software organizations may have to begin reassessing their business models. Organizations may decide to continue to operate as nonprofits while paying federal income taxes, generate and rely on profits, or adopt a middle path between those two extremes.

Nonprofit Corporations

Some open source software projects may decide to continue to operate as nonprofit corporations even if the organizations have to pay federal income taxes as long as their federal income tax burden is not excessive. Organizations form as nonprofit corporations at the state level.[28] This process of entity formation occurs before and independent from any decision regarding exemption from federal income taxes.[29]

Depending on the laws of the state of incorporation, a nonprofit corporation may or may not be exempt from state sales, property, and income tax.[30] For example, a Massachusetts nonprofit corporation may qualify for state tax exemptions only after receiving federal income tax exemption.[31] Once the Massachusetts nonprofit corporation is exempt from federal income taxes, it must apply separately for exemption from state sales, property, and income taxes.[32] An open source software organization interested in continuing to operate as nonprofit corporations should consider the laws of the state in which the organization is incorporated in conjunction with its own financial situation.

An organization may find, for example, that its operating profit is so low that paying federal and state taxes does not impose an undue burden on the organization.[33] In this situation, the organization may elect to operate as a nonprofit corporation and would not need to change its entity type or business model. On the other hand, if the organization found that the resulting tax burden was excessive, it may need to consider adopting a new business model.

For-Profit Entities: Corporations & Limited Liability Companies

For open source software organizations that consider adopting a for-profit business model, three primary entity forms are available: C corporations, S corporations, and limited liability companies.

C Corporations

The C corporation is the default entity type for most businesses that seek outside investors, particularly in the venture capital arena.[34] Corporations offer a few benefits that have helped them retain relevance as new entity forms have developed. First, corporations limit shareholders' liability for corporate debts.[35] Shareholders' liability may not exceed the capital they have invested in the corporation.[36] Additionally, outside investors often prefer to invest in corporations because state corporate law is extensive and corporate case law is well settled.[37] Businesses and shareholders can thus better predict the potential legal consequences of a business decision and act with greater certainty. C corporations also have some flexibility in structuring their ownership. This flexibility can allow C corporations to restrict voting rights to certain classes of shareholders or offer preferred shares that afford investors precedence in receiving dividends over common stock holders.[38] A C corporation may have unlimited shareholders and multiple classes of shareholders with different rights.[39]

C corporation tax status offers both advantages and disadvantages. In addition to offering limited liability, arguably the most well-known trait of C corporations is double taxation. Corporate earnings are taxed twice—once at the corporate level and once at the shareholder level, when the corporation makes distributions to its shareholders.[40] Additionally, corporate losses do not pass through to shareholders, so shareholders cannot use corporate losses to offset their own capital gains.[41]

On the other hand, the double taxation structure may benefit some C corporations when compared to the pass-through structure of S corporations (discussed in the next section). Unlike an S corporation – in which corporate in-

come passes through the corporate structure directly to shareholders – a C corporation retains all earnings not distributed as dividends.[42] As a result, shareholders in C corporations do not pressure the corporation to distribute earnings immediately, as shareholders in S corporations might.[43] C corporations, therefore, can retain and reinvest their earnings more easily than S corporations.[44]

Furthermore, while S Corporations have to elect S Corporation status and meet IRS eligibility requirements, C corporations retain a default IRS status.[45] C corporations do not have to exert any extra effort to obtain their tax status and do not run the risk of losing it.[46] This default position provides greater tax certainty for the corporation and its shareholders.

C corporations' corporate and tax traits make them a good choice for open source software projects that wish to operate on a for-profit business model and solicit investments from outside investors. The relative predictability of C corporations makes them ideal for attracting angel investors and venture capital. On the other hand, C corporations probably would not appeal to projects that wish to limit ownership to specific investors who hope to gain pass-through tax benefits or structure the organization beyond the confines of settled corporate law.

S Corporations

An organization can become an S corporation by incorporating at the state level and then electing to be taxed as an S corporation at the federal level.[47] Because S corporations are simply corporations at the state level, they share many characteristics with C corporations. For instance, S corporations also offer limited liability to shareholders and are subject to the same laws and case law at the state level.[48] Unlike C corporations, however, S corporations may only offer a single class of stock and may not have more than 100 shareholders.[49] Partnerships, corporations, and non-resident aliens may not hold shares of S corporations,[50] limiting the pool of potential investors for an S corporation. On the other hand, S corporations' gains and losses pass through directly to shareholders, allowing S corporations to avoid double taxation.[51] Certain investors may prefer these pass-through benefits to the certainty attached to C corporation status, as discussed above. S corporation status may be ideal for projects that want to limit ownership to a small group of U.S. resident individuals who prefer S corporations' pass-through characteristics.

Limited Liability Companies

Limited liability companies ("LLCs") offer the owners flexibility to create their own rules for the LLC via contract while limiting their liability. Similar to shareholders of corporations, shareholders in LLCs (generally called "members") "are not personally liable for the entity's debts and obligations."[52]

LLCs' governance and operations are largely determined by an LLC operating agreement signed by the LLC's members.[53] For example, the operating agreement may delegate management responsibilities to a certain member or members, or it may spread management responsibilities among all members.[54] Because members can customize their LLC through contract, and because the LLC form is a more recent creation, LLCs do not benefit from the same legal certainty as corporations.[55] The legal consequences of an LLC's actions are more difficult to predict than those of a corporation, and in certain instances this uncertainty could have a chilling effect on the LLC's willingness to take risks. Like S corporations, LLCs offer pass-through tax benefits to their members, although an LLC also may elect to be taxed as a C corporation.[56]

Organizations with owners who wish to construct their relationship through contract, do not mind the greater legal uncertainty attached to LLCs, and want pass-through tax benefits may consider forming as an LLC. Organizations that wish to seek funding from outside investors who are more comfortable with the familiarity of corporations, however, may be better served by another type of entity.

A Third Way: Benefit Corporations

Benefit corporations offer many of the same benefits as C corporations while also elevating organizations' purpose-driven missions. Benefit corporations were created to address a gap in the framework of legal entities. Traditionally, U.S. federal and state law created a binary system of entity formation: entities focused on either generating profit or promoting a non-profit,

purpose-driven mission.[57] Directors of corporations owe duties to their corporations to act in the corporations' best interests.[58] For for-profit corporations, case law generally focuses on whether the director acted in the corporation's financial best interests.[59] No clear authority empowers for-profit corporation directors to consider other interests; for instance, while a director may be aware of the effects of a business decision on workers, customers, or other stakeholders, neither statutes and case law state that a for-profit corporation director can consider these interests on the same level as, or instead of, the corporation's financial interests.[60]

Unlike traditional corporations, benefit corporations can balance financial and non-financial interests in making business decisions. Under model legislation passed in several states, benefit corporations have a "general public benefit" purpose and also may identify "specific public benefit" purposes.[61] Considered to be "in the best interests of the benefit corporation," these mission-based purposes are on equal footing with the corporation's for-profit purpose.[62] In making a business decision, directors may weigh the corporation's mission-based purposes as well as the decision's potential effects on stakeholders, the greater community in which the benefit corporation operates, and the environment.[63]

Mission-driven open source software initiatives may want to consider incorporating as benefit corporations so that they can place their mission-based purpose front and center in decision-making. Benefit corporations offer opportunities and disadvantages for raising capital when compared to traditional for-profit C corporations. Because benefit corporations are structured and operate similarly to C corporations, many investors may be attracted to benefit corporations by the familiarity of their structure.[64] Investor interest may increase due to benefit corporations' mission-driven characteristics, particularly if investors share the corporation's values or goals. On the other hand, benefit corporations may have difficulty attracting funding from risk-averse investors who are wary of the challenge of balancing financial and non-financial interests and of the lack of case law relating to benefit corporations' business decisions. Benefit corporations also share C corporations' disadvantages, including double taxation. Certain investors may prefer another corporate structure. However, if an organization wishes to maintain the C corporation's broad appeal to investors while also making business decisions in line with the organization's social mission, the organization may wish to consider incorporating as a benefit corporation.

Conclusion

Open source software organizations seeking to move beyond 501(c)(3) nonprofit status may choose from a plethora of corporate forms and tax designations. 501(c)(4) and 501(c)(6) designation may offer a small number of open source software projects and umbrella organizations continuing access to exemption from federal income tax. Otherwise, organizations may decide to continue to operate as nonprofits without receiving tax-exempt status.

Other organizations may adopt a for-profit business model and choose from among C corporations, S corporations, and LLCs, depending on each organization's goals, need for outside funding, and management preferences.

Finally, organizations that wish to generate profits while also furthering a social mission may consider incorporating as a benefit corporation. As the open source software community begins to move beyond 501(c)(3), organizations may move beyond a one-size-fits-all model and increasingly explore and experiment with new corporate structures that best fit their needs and goals.

PART II

GOVERNANCE MODELS FROM THE OPEN SOURCE WORLD & BEYOND

Introduction

Because of the importance of community participation and contribution in the open source world, some projects believe that formalized structures may hinder their growth. They worry developers will perceive formal governance as "red tape" and be less willing to contribute.[65] Certain open source founders also fear that after implementing a governance structure, they will lose control over the direction of the project.[66]

These concerns can be dispelled by carefully selecting an appropriate governance model. A properly executed governance structure should encourage developers to contribute, not hinder contributions. Further, founders can retain as much control over projects as they want by choosing a governance structure in line with their goals and needs.[67]

This section of the report examines several examples of governance models, including benevolent dictatorship, meritocracy (sometimes known as enlightened dictatorship), delegated governance, dynamic governance, and a federated model. It explores the skills and traits project leaders may wish to develop or identify in others to best further the goals of the project. In highlighting known and more novel governance models, the authors hope to help fledging groups make informed choices so that they may develop the community and practices they envision.[68]

Levels of Control and Openness

Before an open source project chooses a specific governance model, it may be useful to ask two questions: (1) how open to new contributors should the project be; and (2) how much should project processes be predetermined? The ways in which a given open source initiative answers these two questions can have significant impacts on the project.

Openness: The Cathedral & Bazaar

A chief tenet of open source software development is that anyone can contribute to a project. But projects define "anyone" in different ways. In the early days of large-scale open-source collaboration online, Eric Raymond characterized the two ends of the spectrum of open software development as the "cathedral" and the "bazaar."[69] Even today, open source organizations generally fall nearer to one end of this spectrum or the other.[70]

In a cathedral model, a small team of dedicated contributors builds and maintains the open source project.[71] Usually, some connection to the project or demonstration of skill is necessary to participate, and product releases are closely controlled by the project leadership.[72] As a result, cathedral projects involve fewer people overall, and their releases happen less frequently.[73] In the earliest days of open source software development, before the Internet was used widely outside of universities, the cathedral model was the primary means by which open source software developed.[74]

On the opposite end of the spectrum is the bazaar model. The term was coined to describe the processes at Linux, one of the first open source projects to eschew the cathedral approach.[75] True bazaars are open to all contributors and harness the power of many part-time developers who work collectively via the Internet alone. Rather than closely monitoring every step in the development process, a bazaar gives great autonomy to its large base of contributors and generally puts out frequent releases, even if they are not perfectly honed, to correct bugs and increase the sense of accomplishment among the dispersed developers.[76]

Today, many open source projects fall somewhere between the bazaar and the cathedral and sometimes shift as priorities and capacities change.[77] Evaluating this spectrum can help new project leaders envision the kind of collaborative ethos they would like to create for their open-source community.

Control in Open-Source Organizations

Similarly, open-source projects run along a spectrum vis a vis how much formal control exists in the organization's governance structure. Di Tullio and Staples, two management information systems researchers, ran analyses of a sample of open source software organizations with established governance structures to better understand the existing models in the community.[78] The researchers established several categories of "governance dimensions" within these projects: ownership of assets, establishment of project goals, community management, software development processes, conflict resolution, use of information, and use of tools.[79]

Table 3. Open Source Project Governance Configurations: Results from Cluster Analysis

Governance dimensions	Cluster 1 Open community (*n* = 98)	Cluster 2 Authoritarian community (*n* = 58)	Cluster 3 Defined community (*n* = 28)
Ownership of assets	Highly restrictive	Highly restrictive	Highly restrictive
Chartering the project	Community goals	Project goals	Community goals
Community management	Decentralized management	Centralized management	Decentralized management
Software development processes	Undefined process	Somewhat defined process	Managed process
Conflict resolution	Unmanaged conflict	Unmanaged conflict	Managed conflict
Use of information	Clear and enforced	Unclear and not enforced	Clear and enforced
Use of tools	Clear and enforced	Unclear and not enforced	Clear and enforced

Figure 1: 65, from Dany Di Tullio & D. Sandy Staples (2013) The Governance and Management of Open Source Software Projects, Journal of Management Information Systems, 30:3, 49-80, http://dx.doi.org/10.2753/MIS0742-1222300303

The organizations in their samples were then grouped into three clusters based on their scores in these categories, as shown in Figure 1: open community, authoritarian community, and defined community. The three organizational clusters varied in how much they defined process in development and community management.[80]

As the name indicates, "defined communities" had managed processes for every function of their organization, and the community as a whole managed the project.[81] It was these communities that Di Tullio and Staples found "had the most positive [community] climate and the most effective coordination," while the authoritarian community had the poorest results in both metrics, and the open community model fell somewhere in between.[82] The researchers suspected the defined community model was most successful in these areas because it had clearly defined community roles, instructions, and project.[83] The defined community model created "a balance between freedom [for community input] and control" over project processes.[84] As a result, contributors felt involved in the project while also understanding how they were expected to contribute. Di Tullio and Staples' research underscores the importance of clarity in process and roles when designing an open-source project's governance structure.

With that general framework, the following sections examine several governance models to help open source projects create their own versions of defined communities.[85]

Model A: Benevolent Dictatorship

Introduction

In a benevolent dictatorship, one or a few project founders are the final arbiters and ultimate decision-makers for all aspects of an open source project.[86] An open source organization may decide to adopt a benevolent dictatorship structure when the project's founder wants to keep control of the project over its lifespan.[87] The founder may want decisions to happen through him or her – with decisions made either by the founder or with the founder's direction. The founder may also may desire to be in charge of a body of people who make decisions about the project. Given this powerful position, a founder operating as benevolent dictator must have deep knowledge about all aspects of the project and skill and expertise in building community and trust.

Despite this concentrated authority, a benevolent dictatorship is not a one-man show. Just like all open source software, projects governed by benevolent dictatorships rely on contributors to make low-level decisions and encourage participation in discussion on project direction.[88]

However, final decisions are made by the benevolent dictator and not by the community as a

whole. The strategy and direction of a project, while influenced by contributors, is subject to a final decision by the benevolent dictator (or those the benevolent dictator appoints).[89]

Roles

There are multiple roles in a benevolent dictatorship. Most importantly, the benevolent dictator leads the technical direction of the project and organizes the community.[90] Usually, a benevolent dictator is self-appointed because of the way open-source organizations begin and expand. If more than one or two people found a project, one of the original developers will sometimes declare him or herself a benevolent dictator in order to establish a clear chain of command.

Secondly, there are committers. Committers not only contribute code to the project but also screen the contributions of others – an extremely important task.[91] In most cases, the benevolent dictator "taps" a respected member of the open source community to become a committer.[92] Committers often have only informal control over a certain area of the project and hold no more sway in the decision making process; however, due to his or her respected position, the benevolent dictator is more likely to listen to a committer's suggestions.[93] Usually, committers have direct access to the code repository.[94]

A third role is that of contributor. Contributors are active members of the community who make contributions to the project but usually do not have direct access to the code.[95] There is no formal process to becoming a contributor; a community member must merely contribute in a beneficial way to the project.

Conflict Resolution

Little exists in the way of a formal conflict resolution process in most benevolent dictatorships. Because the dictator has the final say on all decisions, disagreements can be handled informally within project teams.[96] That being said, the dictator may not always intervene when conflict arises, choosing to remain on the sidelines as the community sorts out the problem, or until it asks for the dictator's opinion.[97]

Key Elements of the Benevolent Dictatorship Model for Any Open Source Initiative

Authority is centralized at the top

The benevolent dictator ultimately gets to call the shots in the project and retains control of the direction and scope. He or she sets the general course, and the community follows this lead (though most benevolent dictatorships give contributors room for creativity). The chain of command is clear to all involved.

The community perception of the dictator can impact the success of the project

No one wants to work with a difficult leader, especially if the work is voluntary. While the benevolent dictator often garners a lot of inherent respect for his or her technical skills and vision, instances of poor leadership or disrespectful behavior can alienate community members enough to abandon ship. Successful benevolent dictators must be judicious about using their trump card and anticipate backlash from those who invest time and effort into a patch only to see it rejected.

Individual assignments are ad hoc and functionality decisions are usually dependent on the benevolent dictator

Whereas some of the governance models addressed below are more strategic about placing contributors, the benevolent dictator model does not require any formal assignment of tasks. The benevolent dictator may assign certain committers to specific tasks, but all contributors are generally free to work on whatever they would like. Benevolent dictatorships are very often good examples of the open bazaar at work in open source governance. Many contributors will appreciate this freedom. But, the potential for friction between the contributor community and the leadership increases if contributors devote time to an aspect of the project that is eventually rejected or dismissed as non-essential. A benevolent dictator therefore needs to make her vision and priorities very clear to the community at large so that work can progress smoothly.

Governance In Practice: Linux as Benevolent Dictatorship

In 1991, Linus Torvalds made available a Unix-like kernel on the Internet for free.[98] He invited developers from anywhere to contribute code, and within two months, Linux 1.0 was created.[99] Since then, thousands of developers have contributed to the software and Linux has become one of the most powerful operating systems in the world.[100]

Linux has no formal hierarchy that determines which members should contribute to each project or task.[101] Any person can contribute to issues about which he is passionate, although another developer will improve her work if it is inadequate.[102] Linus Torvalds's authority is restricted to having the final word on what to do with patches provided by community members.[103]

Despite the fact that Torvalds makes overall final decisions, he does not wield totalitarian power.[104] Most communication about technical decisions occurs through the Linux mailing list and Torvalds has to justify all his decisions based on technical arguments.[105] Accountability is essential, and only by earning the community's respect can Torvalds's leadership be maintained.[106] Further, Torvalds does not personally manage the whole kernel; instead, he has designated a few members, whom he calls "trusted lieutenants," to lead certain subsections of the code.[107] If Torvalds disagrees with a decision made by one of his lieutenants, however, he may overrule the lieutenant.[108]

Torvalds's leadership style is not always popular in the Linux community.[109] Because Torvalds can reject a patch in which many developers invested a large amount of time and effort, his decisions sometimes cause tension.[110] One user wrote a long email about Torvalds's faults, beginning with the following statement:

> Linus doesn't scale, and his current way of coping is to silently drop the vast majority of patches submitted to him onto the floor. Most of the time there is no judgement [*sic*] involved when this code gets dropped. Patches that fix compile errors get dropped. Code from subsystem maintainers that Linus himself designated gets dropped. A build of the tree now spits out numerous easily fixable warnings, when at one time it was warning-free. Finished code regularly goes unintegrated for months at a time, being repeatedly resynced and re-diffed against new trees until the code's maintainer gets sick of it. This is extremely frustrating to developers, users, and vendors, and is burning out the maintainers. It is a huge source of unnecessary work. The situation needs to be resolved. Fast.[111]

Despite the fact that Torvalds is not always a favorite in the community, most contributors respect his technical expertise enough to continue developing for the project. Linux thus remains quite popular and successful.

Model B: Meritocracy

Introduction

A meritocracy (also called an enlightened dictatorship) governance structure is loosely organized and rewards participants who make valuable additions to the project.[112] All members generally are on equal footing. But, an individual's standing can be enhanced by "merit," or by contributing meaningfully over a period of time.[113] Most decisions about the project are made by the community as a whole, which develops its own collective ethos on what contributions and behaviors are acceptable and usually are more bazaar-like in operation.[114] Meritocracies are appropriate for open source projects whose founders believe that the community as a whole is best equipped to make decisions about the direction of a project. In these cases, a formal decision-making and contribution structure is essential.

Roles

In meritocracies, there are contributors, committers, and a small project management body (usually a committee).[115] Contributors are developers who contribute to a project in any meaningful way. Usually, this means contributions of code, but can also include updating documentation and supporting new users.[116]

Committers are community members who have shown their dedication to the project through

ongoing engagement with the community. As a result, they can make direct changes to the code and do not have to contribute improvements through patches.[117] Committers also look at contributors' code and judge what should and should not be included. Committers themselves do not technically have more authority than contributors but undergo a different review processes. The work of committers is judged by the community as a whole, when the community decides what should be allowed in the official release.[118]

The largest difference between committers and contributors is apparent when the code is reviewed. When a contributor submits code, it is through a patch or pull request, which waits for committer approval before entering the code base.[119] When a committer submits code, the committer modifies the code base directly and notifies relevant developers of the change. This gives those developers a chance to review code and suggest modifications, known as a "commit-then-review" process.[120]

A meritocracy is generally run by a project management committee. Members of this committee are in charge of making sure projects run smoothly and efficiently. They review code contributions, participate in strategic planning, approve changes to the governance model and manage legal issues (such as code licensing policies) within the project.[121] New members of the committee are invited to join by existing members, and the original make-up is chosen by the project creators.[122] Though committee members have no significant authority over other community members, the project management committee does select committers. The committee also makes decisions when the community cannot reach an agreement.[123]

Conflict Resolution

In meritocracies, decision making usually involves a proposal, a discussion, a vote (if no agreement is reached during the discussion), and – finally – a decision. Any community member can propose a change by submitting a patch or proposal through project communication channels.[124] The proposed change is reviewed and discussed by the community. In general, if no one explicitly opposes a proposed change, it is assumed to be approved –a process called "lazy consensus."[125] The "lazy consensus" reduces time wasted by forcing people to explicitly agree with new contributions, and instead only requires feedback from opponents. If there is opposition to a proposed change, the change is discussed and voted upon, and a decision is made.[126]

Key Elements of the Meritocracy Model for Any Open Source Initiative

Authority is decentralized

Though the original project founders may serve on the project management committee, which leads discussion or resolves conflict on big-picture community issues, the general direction of the project is ultimately set by the community at large.

Committers play a unique role in shaping the project

Because committers get to approve contributors' patches and their own contributions are reviewed after the fact, they may end up having a larger say in how the project progresses. Sometimes, this power means that meritocratic organizations end up overlooking useful contributions because of committer biases. For example, researchers examining GitHub pull requests found that women's pull requests were incorporated into projects nearly ten percentage points less often than men's, even though women who used non-gendered screennames had the best commit rates for contributions.[127] Good committers, therefore, need to be the most cognizant of community values and able to catch internal bias.

Community norms are essential

Because a meritocratic project will follow the community's desires, initial project members must work diligently to create consensus on what is acceptable in the community, as well as what contributions or traits are valued. Defining "merit" is particularly imperative since both patch commits and becoming a committer depend on demonstrations of bona fide "merit." Successful meritocracies often include non-technical skills when considering a contributor's "merit." As evidenced in the case study below, Apache has worked to clearly define merit for their community, explicitly including non-coding contributions in its parameters.

Governance in Practice: Apache as Meritocracy

Apache is an open source organization run through a meritocracy.[128] In 1999, a group of people that called themselves the "Apache Group" and had organized themselves together several years earlier, came together again to continue to support and maintain the HyperText Transfer Protocol Daemon (or "HTTPD") web server written by the National Center for Supercomputing Applications.[129] The original creators of the HTTPD web server had abandoned the project, leaving the users with no support.[130]

These users came together and started developing patches and information on how to fix problems.[131] One user even created a mailing list to encourage collaboration within the group.[132] As the Internet grew larger, the group expanded, and Apache also grew.[133] With its growth, Apache needed a formalized structure for contributions and decision making.[7] Apache chose a meritocracy largely because of its origin as a large group with a common interest in supporting HTTPD. There was no one original "founder" who might want to establish a benevolent dictatorship, or other similar technical structure. As the group grew larger, more and more developers contributed to the project, and more people started using it and wanting to contribute.

At first, users would only contribute small changes, such as by making minor patches or replying to emails on the user list. When the group decided a user had established enough "merit" to be trusted with more responsibility, it granted the user direct access to the code repository. As the group grew even bigger, it formalized its governance into what it is today. If a contributor consistently participates in a meaningful way, he or she can become a committer. Certain committers later become "members," the Apache designation for participants in the main project management committee, which gives them a role in the foundation and a voice in certain organization-wide votes.[134]

Apache has scaled to become a large and successful organization because newcomers are considered volunteers who want to help (rather than competing for the limited power and resources in the organization). At the same time, Apache – like other meritocratic organizations – sometimes struggles with questions about process and conflict resolution because of the inherent power of committers and its minimal conflict resolution strategies.[135] Apache makes efforts to combat this by publicly stating that they "value the community more than the code," and providing community building resources, such as a mentorship program for new contributors.[136] Apache also explicitly states that one can become a committer through dedication to community building and product management, not only code contributions.[137]

Model C: Delegated Governance

Introduction

When an open-source project's founders want to allow the community to take ownership and leadership in the project while establishing a more regulated set of processes for its progression, they may turn to a delegated governance structure. In a delegated governance, a body of leaders, sometimes known as the "Community Council," is chosen or elected to oversee the project, resolve conflicts within the community, modify the community norms and processes, and determine the project's core values.[138] Depending on the size of the project, a delegated governance system may have several sub-councils that manage specific aspects of the project or community and report back to the main Community Council. Like a meritocracy, the delegated governance model involves more contributors in governance than a benevolent dictatorship. But, it also has a clear hierarchical structure and a designated set of leaders.

Roles

Like meritocracies and benevolent dictatorships, delegated governance involves regular community contributors and committers who help manage the approval process for patches. The key difference between meritocracy and delegated governance is the addition of councils, which do have authority over the direction of work and the management of the community. Council members may be chosen initially by the founder of the project but afterward elected through a community-wide vote.[139]

The size and responsibilities of the council will depend on the organization. Councils are generally responsible for overseeing the issues they are tasked to cover, and discussing them and voting when questions arise. Normally, these positions are held for terms of a few years, and some organizations enforce term limits to ensure community members have ample opportunity to circulate into leadership roles.[140] If an organization is large enough, different subcouncils will have authority over areas of the project, from technical direction to community culture to distribution, each run by council members who ideally have relevant expertise.[141] While contributors and committers may navigate small-scale decisions about features or content on their own, larger discussion about the project's overall direction take place in councils.

Conflict Resolution

In the delegated governance model, minor disagreements are managed through a "lazy consensus" model, similar to a meritocracy. When this process fails to resolve the issue, the group in question may choose to escalate the issue by bringing it to the relevant council for discussion and a vote.[142] In an organization with different tiers of councils, a sub-council may vote to escalate the issue up the decision-making ladder based on its applicability to the project at large or the sub-council's inability to reach an agreement. Whereas a pure meritocracy makes these decisions directly or through group deliberation, a delegated governance model uses a representatives to resolve issues.

Within the organization's councils, and particularly the top tier, conflicts may be resolved in different ways depending on the size and make-up of the group. If the council has an even number of members, certain members may receive tie-breaking power.[143] If the project founder holds a place on the Community Council, he or she may have that role. Additionally, some more commercially-focused projects reserve a council seat or seats for employees of for-profit companies that contribute significantly to the open-source project's development, which may impact the democratic nature of project governance.[144]

Key Elements of the Delegated Governance Model for Any Open Source Initiative

Authority is centralized at the top, but also distributed through a chain of command

The delegated governance model has a central seat of power in the Community Council, but its subcouncils get to exercise authority over particular project areas, escalating questions up the chain of command when necessary.

Many community members can hold some form of leadership role

Open source organizations that use delegated governance have more opportunities than a benevolent dictatorship or a meritocracy to recognize community members by giving them leadership positions. This vote of confidence can keep a contributor more invested in the project and allow for greater specialization within the project. On the other hand, more leaders means more deliberation, and conflicts may take longer to resolve than in other forms of governance.

Control over project direction will vary depending on how councils are selected

If a project's founders sit on or appoint the Community Council, it is likely they will retain a certain level of control over the project. If the councils are elected by community members, community representatives will have the ability to exert influence over the project's evolution. In some scenarios, this will mean a nimbler organization that adapts quickly to change, but in others, complete changeover could cause organizational whiplash as priorities change rapidly from one election to the next.

Governance In Practice: Is Ubuntu an Example of Delegated Governance?

Though delegated governance was first conceived and described by Jono Bacon, an open source thought leader and community manager of Ubuntu, there are several ways in which Ubuntu does not embody the definition of delegated governance as Bacon describes it.[145] Founded by software entrepreneur Mark Shuttleworth in 2004, Ubuntu was established early on with the goal of creating an open source Linux distribution that was commercially useful while also allowing for open and transparent governance by the open source developing community.[146] The solution Shuttleworth created is to establish governing councils whose membership is open to all of the Ubuntu contributing community.

Ubuntu has a Community Council that handles all aspects of how the Ubuntu community functions, and a Technical Board that is responsible for all technical policy and process decisions.[147] The figure below highlights the organization's structure:

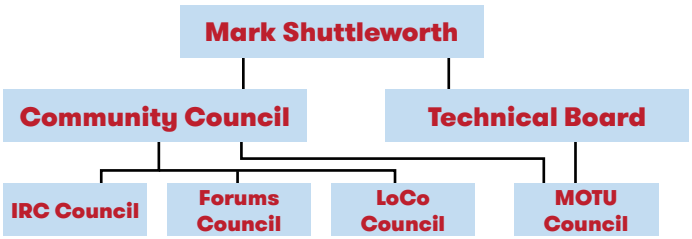


Figure 2: From Jono Bacon's "The Art of Community," chapter 8 "Governance - Ubuntu Governance Example", O'REILLY MEDIA (2009).

The Ubuntu council system establishes a clear path for managing the software's development and maintenance and escalating issues should they arise. What this diagram also reveals, however, is Shuttleworth's role lodged within the delegated governance model. Shuttleworth sits on the Community Council and Technical Board and has tie-breaking power in each, though he cannot override the decisions of either of his own volition. Additionally, Shuttleworth's for-profit company, Canonical, which relies heavily on Ubuntu's product for its own work, has employees who work in the Ubuntu community as part of their role.[148] Finally, Shuttleworth nominates Community Council and Technical Board positions (though any community member can indicate their desire to be nominated), which then go to a vote in the relevant community.[149]

Shuttleworth made these choices intentionally when designing Ubuntu to ensure that the project harnesses the power of its community while keeping in line with the needs of Canonical, and to that end the structure has been effective for Ubuntu's continued development and growth. Given the amount of influence Shuttleworth and Canonical wield over the project, some may feel this makes the governance structure fall less squarely in the delegated governance model. Open-source projects could adapt this model to be more democratic by eliminating the tie-breaking power of a particular board member and creating a more open process for filling out council seats, such as an election.

Model D: Dynamic Governance

Introduction

The dynamic governance model, also known as sociocracy, is one designed to distribute authority to every sector of an organization.[150] Both businesses and nonprofits have used this model to empower employees and remove top-down authority. Because the open source community overall feels strongly about the importance of community input and involvement, this very distributed model may appeal to certain open source initiatives.

Roles

According to experts from the Sociocratic Center of the Netherlands, the defining characteristic of dynamic governance is the "double-linked circle." A dynamically governed organization is divided into different circles of participants, each with specific responsibilities.[151] In most dynamic governance models, all members of an organization sit on one of its circles, which deal with a specific area of the organization, much like subcouncils in a meritocracy.[152] Circles are organized hierarchically, culminating with the "general circle" (where executive leaders sit), but each circle membership also includes two representatives from the circle below (if there is one) and the circle above.[153]

This "double link" connects the leader of the circle (a team manager, for example) to the group higher up, as well as an elected member who does not hold a specific leadership position.[154] Within any particular circle, members deliber-

ate and reach consensus on any decisions regarding working process and policies.[155] While the circle's participants handle long-term direction or strategy questions through regular meetings and discussions, the manager or team leader organizes day-to-day activities, assigning particular circle members to tasks in accordance with the circle and organization's larger goals.[156] The deliberative process within a participant circle ideally follows this model: group discussion on a topic to identify a solution and people to implement it; enactment of the solution and monitoring the results; and measuring the impact and soliciting group feedback to determine next steps.[157]

Dynamic governance is well suited for an open source organization committed to keeping the average contributor as engaged in the workings of the project as possible because it traditionally necessitates a certain level of commitment from all levels, including volunteers. In the same way, every contributor in an open-source project would ideally be part of a circle, involving that contributor in the leadership structure.

Given that open source contributors may come from all over the world and may not be able to meet live consistently for a circle, project founders could modify this aspect of the model to find the level of project involvement best for their community. Projects can organize circles around specific technical deliverables or community management needs that are double-linked to the general circle, where project founders may sit. Some dynamic governance models include external experts or stakeholders in the general circle, which some open-source projects might find is a useful way to bring a diversity of skills into the organization.[158]

Conflict Resolution

Consensus is the means of resolving all disagreements in the dynamic governance model. When a circle member identifies a problem or question, the group convenes to discuss it, allowing all circle members an opportunity to voice their opinions and ask questions.[159] In the world of dynamic governance, a circle arrives at consensus when none of its members still hold a reasonable objection to the proposed solution.[160] The requirements of consensus do not mean that a circle cannot move forward until one side sways the other to abandon their argument; a circle may choose an action that allows the group to get more information about the problem, seek outside guidance, or ask a higher-up circle to weigh in.

Key Elements of the Dynamic Governance Model for Any Open Source Initiative

Decision-making power is highly dispersed throughout the community

In a dynamic governance model, every contributor will have a voice in how the project functions, even if some ideas come from the top down, which can create a strong sense of belonging and empowerment for all. This model does, however, require contributors to commit to being involved consistently, while other governance structures allow the average contributor to flit in and out, handling discrete tasks but staying unengaged with the organization itself. Certain contributors will find this level of commitment more appealing than others.

Need for consistency may limit community growth potential

Because every contributor would be part of a project circle in an open source project with a dynamic governance structure, it may be challenging to add new contributors as soon as they show up, or to add a lot very quickly. For projects that expect to expand rapidly, dynamic governance may not be a sustainable model, as it lends itself more to a cathedral than a bazaar. Small- to mid-sized projects that want to collaborate closely with all contributors will find dynamic governance a better fit.

Understanding and acceptance of deliberative processes and governance structure at all levels is essential

One risk of a decentralized model is that, lacking a strong central authority, smaller decisions get drawn out and institutional hang-ups will derail work toward the organization's goals. To avoid this pitfall, open source leaders must do their best to educate the community about how the dynamic governance model should work within the organization and to establish very clear procedures for how the circle model should run. It may also be helpful to identify the ideal size of each circle and develop a process for funneling new contributors to the areas of greatest need.

All About Boards: Examples from the Nonprofit World

Introduction

A traditional, formal nonprofit is legally obligated to have a board of directors charged with ensuring "prudent use of all assets," adherence to applicable laws and ethical mandates, and collective decision-making "in the best interest of the nonprofit corporation."[161] But these boards often take on a role in organizational governance beyond their legal obligations, and many unincorporated or all-volunteer organizations rely on the board structure for leadership and volunteer management. Open source organizations could use lessons from the nonprofit world to leverage the board of directors model, even if they are not a nonprofit, as a stand-alone governance model, as a feature of any of the traditional open-source governance models we have already discussed, or as a separate "big-picture" evaluative body. Additionally, recommendations about board selection and membership are also applicable to an open-source project looking to fill leadership roles.

The essential purpose of a board in a nonprofit or mission-oriented organization is to advise and steer the direction and long-term vision of group. As an entity, the board can accomplish this in a number of ways. In volunteer-run organizations, the board will act as the staff, creating and maintaining organizational documents, galvanizing and assigning volunteers to specific tasks, procuring funding, and other tasks as needed.[162] In a publication on all-volunteer organization boards, the National Center for Nonprofit Boards and the Support Center for Nonprofit Management also highlight the board's role in recruiting new volunteer leaders and collectively working to evolve the organization, bringing in outside help if necessary.[163] The most effective all-volunteer boards members also regularly confer with the community they represent, making sure that the board as a whole understands community needs and concerns, and also that the community feels heard by those at the helm.[164] To tackle specific aspects of their organization's work, board members may oversee committees tasked with different aspects of the organization's work, similar to the sub-council system in a delegated governance. A board of directors typically includes roles such as a president and vice president who oversee the entire organization, a secretary who records the deliberations of the group, a treasurer who tracks the organization's finances, and other posts and committees specific to the organization's needs.[165]

Choosing a Board

Nonprofits use a variety of methods to choose board composition. Many nonprofits organize a nominating committee made up of existing board members to identify community members or others who match the board's needs. These committees may also evaluate the current board's efficacy and make suggestions to improve performance.[166] In this model, board members do not always come from within the organization. If the nominating committee identifies a particular institutional need that the current community cannot fill, they will recruit outsiders who have the requisite skills and find the organization's mission compelling.

Other organizations opt for a more democratic model, electing board leadership through a member vote.[167] Some experts in nonprofit management worry that an elected board can become a soapbox for individuals rather than a platform for the organization – an organization's paid leadership may struggle to connect with board members or motivate them to serve the organization rather than themselves.[168] But, for all-volunteer organizations (like many open source projects), an election may help keep members engaged and give recognition to those who are particularly committed.

Notably, organizations need not wed themselves to a particular selection model – some seats on a board might come from a nominating process, while others could be reserved for elected community representatives. One suggestion from McKinsey & Company nonprofit consultants is to establish a two-tiered board.[169] In this format, one tier meets regularly and consists of members who are directly involved with the project, and the other meets annually or semi-annually in an advisory capacity, with members who are experienced but not part of the day-to-day operations of the project.[170]

The suggestions above help illustrate how a board structure can be a helpful format for or supplement to the technical governance of a open-source organization, nonprofit or otherwise. In a small organization, a board can deputize community members and create a clear division of responsibilities, which can expand into councils or other governing sub-bodies over time as the project expands.

A board can also serve as the place where other important but non-development-related work gets done. The recent experience of the io.js community demonstrates how such a format might work. Io.js originated from the existing open-source project node.js (begun by Joyent) because many members felt its organizational and copyright ownership structures posed serious problems.[171] While the new technical governance model spurred higher contributor engagement and a better product, project leaders realized they were neglecting non-technical concerns that might once again cause problems for the project itself if ignored, such as financial planning, legal advising and public relations.[172] As a result, io.js leaders worked with entities at Joyent and the Linux Foundation to create their own foundation and board to supplement their delegated governance technical structure.[173] This format enabled the io.js community to manage technical and organizational well-being needs separately and concurrently. Though io.js chose to go the nonprofit route, other corporate forms could also use a similar vision of a distinct but connected process board and development team to advance their organization.

Federated Nonprofits ("Model E")

One governance model that open-source organizations may want to consider from the nonprofit world is the federated nonprofit (separately described as "Model E" in the accompanying illustrations). The federated nonprofit organizational structure is employed by groups like Girl Scouts of America, the American Red Cross, and other national nonprofits with semi-autonomous regional chapters.[174] These chapters run programs, fundraise, and self-govern for their particular region, often relying heavily on the contributions of volunteers but also coordinate with and take direction from a national governing body.[175]

Generally, regional chapters take direction from the national organization on policies and the services they provide to the community, but make day-to-day decisions on their own and occasionally develop programs specific to their needs. Regional chapters or affiliates have their own leadership teams to run daily operations, supported by the organization's members. To connect the regional and national, many federated nonprofits periodically host national gatherings for members.[176] National boards of directors may be appointed without input from lower-levels, but in some federated nonprofits, members elect the board.[177] In the most participatory federated nonprofits, members will also vote on certain policy decisions or amendments to the organization's bylaws.[178]

Most small- to medium-sized open-source projects would find the federated model too siloed for their organization's needs. To some, semi-independent project chapters might sound like a great way to induce software to fork into discrete projects, something that early-stage organizations generally want to avoid. However, for a larger, better-established open-source organization that works on multiple projects, or for those that create different software versions for specific applications, a federated model would allow these teams to operate independently in their spheres while remaining connected to the top-level leadership through the election of board members.

An open source organization could also choose to give contributors voting rights on certain aspects of project direction at the global level. Whereas in the dynamic governance model, subcouncils are responsible for different aspects of how the project functions, including community and process, a federated model would have each project team be more autonomously responsible for its own community, process, and technical teams. Project chapters could employ a meritocratic, delegated governance, or other governance model within themselves while remaining linked to the umbrella organization.

Norms and Attitudes for a Successful Open Source Software Organization

Regardless of the governance model an organization adopts, community norms and leadership qualities can play a major part in shaping

how the organization operates. Establishing models for leadership and community behavior, either explicitly or through leading by example, can lay the groundwork for strong, productive, and amicable relationships both between leadership members and within the organization as a whole.

One major challenge for all organizations – but particularly those focused on specific skillsets (like institutions engaged in open-source software development) – is recognizing the value of outside perspectives and non-technical skills. Because the heart of every open source project is the development and maintenance of the project's code, it's not always apparent how non-technical contributions should be recognized, or how they might improve development processes or community relations.

In the same vein, those who are the best developers and code contributors may not always possess the soft skills required for open-source leadership roles. Many of the most successful open source companies are ones that recognized this possibility early on and constructed their community intentionally to welcome non-technical contributions and identify community members with leadership potential, whether or not they were the strongest developers.

Apache, as the case study illustrates, is one example where an organization learned to identify non-technical needs and how to fill them. Sometimes Apache taps particular contributors who show a capacity for long-term planning and leadership to step into non-technical roles, while at other times members are recruited and selected solely for having non-technical skills. Consciously selecting for specific non-technical skills and qualities has allowed Apache to create a diversity of backgrounds within the community, which in turn made the organization nimbler and more capable.

Ubuntu's community manager Jono Bacon examined the qualities that make Ubuntu's Community Council work in his book *The Art of Community*. He identifies several traits he thinks all open-source leadership teams should possess.[179] According to Bacon, a good community leader should be, in some combination:

- a good listener, who can make community members feel heard and can use this skill to get a read on the community and resolve issues;
- detail-oriented;
- able to recognize his or her biases and strive to set them aside in order to understand all sides;
- "a fair fighter" who will champion ideas they feel are best for the community while also being willing to concede or compromise for the greater good; and
- reliable, to show up when asked and participate wholeheartedly.[180]

Nonprofits and mission-driven organizations also encounter many of the same issues that open-source organizations face, regardless of mission, including organizing a leadership structure to govern their peers and/or a community they wish to help. Many of these groups are governed by a Chief Executive Officer and volunteer board, but the model of cultivating leadership that they provide could translate well to other governance models.

One suggestion from the nonprofit consulting world is creating opportunities for training in leadership skills for current organizational leadership, or those who wish to join it.[181] Depending on the leadership model, term limits can also help to keep the governing body dynamic and motivated. So can establishing "formal targets" for the leadership's make-up, such as mandating equal gender distribution or a minimum percentage of members in a particular age range or from diverse backgrounds.[182]

Finally, a clear understanding of the project's mission and goals is equally as important in cultivating group leadership and a governance structure as it is in determining a formal corporate governance structure. Having a specific vision helps define expectations of those in leadership roles, both in terms of what a leader needs to accomplish and what qualities she should develop. Leaders who understand the mission of an organization are able "to connect [their] actions to the ultimate purpose of the organization," thereby making them more invested and dedicated in furthering the mission and, in turn, motivating the community.[183]

CONCLUSION

When open source creators launch new projects, their primary concerns may be technical. But expanding their focus to the organizational can have enormous benefits. Projects that make thoughtful decisions around corporate formation may streamline dealings with the IRS and also create stable entities that will help them be sustainable and dedicated to their ultimate missions. Projects that spend time considering questions about corporate formation set themselves up to: create a welcoming project community with engaged and invested contributors; and then govern that community effectively.

The formation and governance options in this report may not fit every project's needs and goals. Open source initiatives have a wide array of interests; while they rely on similar tools, they create many different products and rely on different processes in their development activities. The authors encourage open source project creators and contributors to take what resonates from this guide – whether a whole model or a few practices – and use those lessons to shape their organizations as they see fit.

Though it is sometimes overlooked, the history of the open source movement shows us that the projects that defined their corporate structure and governance practices early and concretely set themselves up for success. While some elements of the landscape have changed – most notably the IRS's attitude toward open source projects – the benefits of intentionality remain. By setting the tone early on through decisions on formation and governance, project founders can harness the unique power of open source communities to create sustainable open source projects from which we can all benefit.

Endnotes

1 26 U.S.C. 501(c)(3) (2012).

2 *See* Jim Nelson, *The New 501(c)(3) and the Future of Free Software in the United States*, Yorba Blog Archives (Jun. 30, 2014), http://blogs.gnome.org/jnelson/2014/06/30/the-new-501c3-and-the-future-of-free-software-in-the-united-states/.

3 *See, e.g.*, *The Mozilla Manifesto*, Mozilla Foundation, https://www.mozilla.org/en-US/about/manifesto/ (last visited Dec. 4, 2014) (Mozilla's list of principles includes the following: "Free and open source software promotes the development of the Internet as a public resource."); *The Linux Kernal Organization*, The Linux Kernal Archives, www.kernel.org/nonprofit.html (last visiting Dec. 4, 2014) ("The Linux Kernel Organization [was] established . . . to distribute the Linux kernel and other Open Source software to the public without charge."); *Foundation Project*, The Apache Software Foundation, www.apache.org/foundation (last visited Dec. 4, 2014) ("Apache™ projects deliver enterprise-grade, freely available software products.").

4 *See The Economics of Open Source Donations*, Packt Publishing, https://www.packtpub.com/books/content/economics-open source-donations (last visited Dec. 4, 2014) (describing typical organizational costs for free open source software organizations that are covered by donations).

5 *See* Nelson, *supra* note 2.

6 *See* The Apache Software Foundation, Form 1023: Application for Recognition of Exemption Under Section 501(c)(3) of the Internal Revenue Code (2000), *available at* http://www.apache.org/foundation/records/ASF-1023.pdf; I.R.S. Final Determination Ltr. (Feb. 21, 2001), *available at* http://www.apache.org/foundation/records/ASF-501c3.pdf.

7 *See* Mozilla Foundation, Form 1023: Application for Recognition of Exemption Under Section 501(c)(3) of the Internal Revenue Code (2000), *available at* http://static.mozilla.com/foundation/documents/mf-irs-501c3-application-form-1023.pdf; I.R.S. Final Determination Ltr. (Jun. 17, 2004), *available at* http://static.mozilla.com/foundation/documents/mf-irs-determination-letter.pdf.

8 Richard Fontana et al, Software Freedom L. Ctr, A Legal Issuer Primer for Open Source and Free Software Projects 24 (2008).

9 Robert McMillan, *Open Source Voting Machine Reborn After 6-Year War With IRS*, Wired (Aug. 6, 2013, 6:30 AM) http://www.wired.com/2013/08/osdv/all/.

10 *Id.*

11 *Id.*

12 *Id.*

13 *Id.*

14 *See* Internal Revenue Service, Be on the Look Out List (Aug. 2010); Be on the Look Out List (Nov. 2010); Be on the Look Out List (Feb. 2011); Be on the Look Out List (Feb. 2012); Be on the Look Out List (Jul. 2012) *available at* http://democrats.waysandmeans.house.gov/press-release/new-irs-information-shows-"progressives"-included-bolo-screening-list.

15 Internal Revenue Service, Be on the Look Out List (Feb. 2012), *supra* note 14.

16 Kelly Phillips Erb, *IRS Gets Big Win In Court As Judge Dismisses Tea Party Targeting Cases*, Forbes (Oct. 23, 2014, 5:47 PM), http://www.forbes.com/sites/kellyphillipserb/2014/10/23/irs-gets-big-win-in-court-as-judge-dismisses-tea-party-targeting-cases/.

17 McMillan, *supra* note 9.

18 *See* Karen Copenhaver et al., Black Duck Software, *Open Source Projects and Foundations: A User's Guide* 15–16 (Aug. 2014), *available at* https://www.blackducksoftware.com/files/webmedia/_webinars/08-28-14_OSSFoundations.pdf.

19 26 U.S.C. 501 (2012).

20 Internal Revenue Service, *supra* note 14, at 48.

21 26 U.S.C. 501(c)(4) (2012).

22 Internal Revenue Service, *supra* note 14, at 48.

23 26 U.S.C. 501(c)(6) (2012).

24 Internal Revenue Service, *supra* note 14, at 49.

25 *See Structure and Voting*, Dojo Foundation, http://dojofoundation.org/about/structure (last visited Dec. 5, 2014); *About the Eclipse Foundation*, Eclipse Foundation, http://www.eclipse.org/org/ (last visited Dec. 5, 2014).

26 Mark McLoughlin, *May 11 OpenStack Foundation Board Meeting*, Just Another GNOME Blogs Weblog (May 17, 2014, 2:52 PM), http://blogs.gnome.org/markmc/2014/05/17/may-11-openstack-foundation-board-meeting/.

27 *Id.*

28 Internal Revenue Service, Notice 844: Federal Tax Obligations of Non-Profit Corporations

(Feb. 28, 2013), *available at* http://www.irs.gov/pub/irs-pdf/n844.pdf.

29 *Id.*

30 *Id.*

31 *Nonprofit Corporation Information*, Secretary of the Commonwealth of Massachusetts, http://www.sec.state.ma.us/cor/corpweb/cornp/npinf.htm.

32 *Id.*

33 Comment from Andy Updegrove, legal counsel at the Linux Foundation, during Mishi Choudhary et al., Organizing FOSS Entities, at Software Freedom L. Ctr. 10th Anniversary Conf. (Oct. 31, 2014), *available at* http://moglen.law.columbia.edu/sflc_at_10/2014-10-31-organizing-foss-entities.m4v.

34 *See* Joe Wallin, *12 Reasons For A Startup Not To Be An LLC*, Startup Law Blog (Sept. 30, 2011), http://www.startuplawblog.com/2011/09/30/12-reasons-for-a-startup-not-to-be-an-llc/.

35 1 Jerold Friedland, Tax Planning for Partners, Partnerships, and LLCs § 1.02 (Matthew Bender ed., 2000).

36 *Id.*

37 *Id.*

38 Joe Wallin, *C Corps v. S Corps*, Startup Law Blog (Jun. 9, 2013), http://www.startuplawblog.com/2013/06/09/c-corps-v-s-corps/.

39 *Id.*

40 1 Friedland, Tax Planning for Partners, Partnerships, and LLCs § 1.02, *supra* note 35.

41 *Id.*

42 Jeremy Halpern, Nutter McClennan & Fish LLP, Choosing an Entity for Startups, *available at* http://www.nutter.com/files/Uploads/Documents/Halpern-Choosing-an-Entity-for-Startups.pdf.

43 Wallin, supra note 38.

44 *Id.*

45 *Id.*

46 *Id.*

47 1 Friedland, Tax Planning for Partners, Partnerships, and LLCs § 1.02, *supra* note 35.

48 *Id.*

49 *S Corporations*, Internal Revenue Service, http://www.irs.gov/Businesses/Small-Businesses-&-Self-Employed/S-Corporations.

50 *Id.*

51 *Id.*

52 1 Friedland, Tax Planning for Partners, Partnerships, and LLCs § 1.02, *supra* note 35.

53 *Id.*

54 *Id.*

55 Wallin, *supra* note 34.

56 1 Friedland, Tax Planning for Partners, Partnerships, and LLCs § 1.02, *supra* note 35.

57 William H. Clark et al., The Need and Rationale for the Benefit Corporation: Why It is the Legal Form that Best Addresses the Needs of Social Entrepreneurs, Investors, and, Ultimately, the Public 7 (Nov. 16, 2011).

58 *Id.* at 8.

59 *Id.* at 10.

60 *Id.*

61 *Id.* at 15–16.

62 *Id.* at 16.

63 *Id.* at 17.

64 *Id.* at 28.

65 *See Governance Models*, OSS Watch (2014), http://oss-watch.ac.uk/resources/governancemodels (last visited Dec. 5, 2014) (discussing barriers to governance models

66 *Id.*

67 *Id.*

68 In the course of our discussion, we will sometimes refer to the size of an open-source project to explain how a governance model might work when involving a certain number of contributors. For the purposes of this guide, we use "small" to refer to a project with less than 20 contributors, "medium" to refer to those with between 20 and 100 contributors, and large to refer to those with more than 100.

69 Eric Steven Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Visionary*, O'Reilly Media, Inc. (1999) https://books.google.com/books/about/The_Cathedral_the_Bazaar.html?id=F6qgFtLwpJgC&printsec=frontcover&source=kp_read_button&hl=en#v=onepage&q&f=false.

70 *Id.*, OSS WATCH

71 *Id.*

72 *Id.*

73 *Id.*

74 *Id.*. Raymond.

75 *Id.*

76 *Id.*

77 *Id.*, OSS WATCH.

78 Dany Di Tullio & D. Sandy Staples (2013) The Governance and Control of
Open Source Software Projects, Journal of Management Information Systems, 30:3, 49-80, http://dx.doi.org/10.2753/MIS0742-1222300303

79 *Id.*, 65.

80 *Id.*

81 *Id.*

82 *Id.*, 68.
83 *Id.*
84 *Id.*, 72.
85 Illustrations of each of the models addressed herein accompany this report.
86 *See Benevolent Dictator Governance Model*, OSS WATCH, http://oss-watch.ac.uk/resources/benevolentdictatorgovernancemodel (last visited Dec. 5, 2014); *see also* KARL FOGEL, PRODUCING OPEN SOURCE SOFTWARE, ch. 4, *available at* http://producingoss.com/html-chunk/social-infrastructure.html#benevolent-dictator-qualifications; *Benevolent Dictator*, P2P FOUNDATION, http://p2pfoundation.net/Benevolent_Dictator (last visited Dec. 5, 2014); Randy Fay, *How Do Open Source Communities Govern Themselves?*, RANDYFAY.COM (March 5, 2012, 8:41 PM), http://randyfay.com/content/how-do-open-source-communities-govern-themselves (listing open-source organizations with benevolent dictatorships).
87 *Id.*, OSS WATCH.
88 *Id.*, Fogel.
89 *Id.*, OSS WATCH
90 *Id.*
91 *Id.*
92 *Benevolent Dictator Model*, OUTERCURVE FOUNDATION, (2014), http://www.outercurve.org/resources/governance/benevolent_dictator.html.
93 *Id.*
94 *Id.*
95 *Id.*, OSS WATCH
96 *Id.*, OSS WATCH.
97 *Id.*, OSS WATCH
98 *Linux – Governance*, P2P FOUNDATION, http://p2pfoundation.net/Linux_-_Governance (last visited Dec. 5, 2014).
99 *Id.*
100 *Id.*
101 *Id.*
102 *Id.*
103 *Id.*
104 *Id.*
105 *Id.*
106 *Id.*
107 *Id.*
108 *Id.*
109 *Id.*
110 *Id.*
111 *Id.*
112 *See Meritocratic Governance Models*, OSS WATCH, http://oss-watch.ac.uk/resources/meritocraticgovernancemodel (last visited Dec. 5, 2014). (discussing the meritocratic governance model in depth); *see also* Ross Gardler, *Meritocratic Governance Models*, OUTERCURVE FOUNDATION (Sept. 25, 2012), http://www.outercurve.org/blog/2012/09/25/Meritocratic-Governance-Models-by-guest-blogger-Ro/; Bacon, Jono *The Art of Community*, O'REILLY MEDIA, INC., (2009)
113 *Id.*, OSS WATCH
114 *Id.*, Bacon.
115 *Id.*, OSS WATCH
116 *Id.*
117 *Id.*
118 *Id.*
119 Ross Gardler, *Essential Tools for Running a Community-Led Project*, OSS WATCH, (Sept. 9, 2013) http://oss-watch.ac.uk/resources/communitytools.
120 *Id.*
121 *Id.*, OSS WATCH
122 *Id.*
123 *Id.*
124 *Id.*
125 *Id.*
126 *Id.*
127 Terrell J, Kofink A, Middleton J, Rainear C, Murphy-Hill E, Parnin C. (2016) Gender bias in open source: Pull request acceptance of women versus men. PeerJ PrePrints 4:e1733v1 https://doi.org/10.7287/peerj.preprints.1733v1
128 *How the ASF Works*, THE APACHE SOFTWARE FOUNDATION, http://www.apache.org/foundation/how-it-works.html (last visited Dec. 5, 2014).
129 *Id.*
130 *Id.*
131 *Id.*
132 *Id.*
133 *Id.*
134 *Id.*
135 Ceki Gülcü, *The Forces and Vulnerabilities of the Apache Model* http://ceki.blogspot.com/2010/05/forces-and-vulnerabilites-of-apache.html (May 21, 2010).
136 *Contributors*, APACHE SOFTWARE FOUNDATION, https://community.apache.org/contributors/ (last visited June 9, 2016); *Mentoring Programme*, APACHE SOFTWARE FOUNDATION, https://community.apache.org/mentoringprogramme.html (last visited June 9, 2016).
137 *Id.*, *Contributors*.
138 Bacon, Jono *The Art of Community*, O'REILLY MEDIA, INC., (2009) http://proquest.sa-

faribooksonline.com.ezp-prod1.hul.harvard.edu/9780596805357/learning_from_the_leaders

139 *Id.*

140 *Id.*

141 *Id.*

142 *Id.*

143 *Id.*

144 *Id.*

145 Bacon, Jono *The Art of Community*, O'REILLY MEDIA, INC., (2009) http://proquest.safaribooksonline.com.ezp-prod1.hul.harvard.edu/9780596805357/learning_from_the_leaders

146 *Id.*

147 *Id.*

148 *Id.*

149 *Governance*, UBUNTU (2016), http://www.ubuntu.com/about/about-ubuntu/governance (last accessed May 5, 2016).

150 John A. Buck & Gerard Endenburg, *The Creative Forces of Self-Organizing*, SOCIOCRACTIC CENTER, http://sociocracyconsulting.com/wp-content/uploads/2015/09/CreativeForces-updated2012.pdf (2012).

151 *Id.*

152 *Id.*

153 John A. Buck & Kerry Koch-Gonzalez, *Dynamic Governance for Nonprofit Organizations*, THE SOCIOCRACY CONSULTING GROUP, http://sociocracyconsulting.com/wp-content/uploads/2014/01/DG-for-Nonprofits-v1-11-14.pdf (2014); Sheella Mierson, *Dynamic Leadership*, THE SOCIOCRACY CONSULTING GROUP, http://sociocracyconsulting.com/wp-content/uploads/2016/02/Dynamic-leadership-v1.1.pdf (2016).

154 *Id.*, Mierson.

155 *Id.*, Buck & Endenburg.

156 *Id.*

157 *Id.*

158 *Id.*, Buck & Koch-Gonzalez

159 *Id.*, Buck & Endenburg

160 *Id.*

161 *Board Roles and Responsibilities*, NATIONAL COUNCIL OF NONPROFITS, https://www.councilofnonprofits.org/tools-resources/board-roles-and-responsibilities (last accessed May 23, 2016).

162 *Why Do You Need a Board?* THE BRIDGESPAN GROUP, http://www.bridgespan.org/Publications-and-Tools/Nonprofit-Boards/Nonprofit-Boards-101/Why-Do-You-Need-a-Board.aspx#.V0N_OTUrK73 (last accessed May 23, 2016).

163 Jan Masaoka *All Hands on Board: The board of directosr in an all-volunteer organization*, THE NATIONAL CENTER FOR NONPROFIT BOARDS, THE SUPPORT CENTER FOR NONPROFIT MANAGEMENT, http://blueavocado.org/sites/default/files/All-Hands-on-Board-3.pdf (1999).

164 *Id.*

165 John Riddle with Tere Drenth, *Managing a Nonprofit*, ADAMS MEDIA CORPORATION (2002).

166 *Finding the Right Board Members for Your Nonprofit*, NATIONAL COUNCIL OF NONPROFITS, https://www.councilofnonprofits.org/tools-resources/finding-the-right-board-members-your-nonprofit (last accessed May 24, 2016).

167 Jan Masaoka, *Boards of All-Volunteer Organizations*, BLUE AVOCADO, http://www.blueavocado.org/content/boards-all-volunteer-organizations (last accessed May 24, 2016).

168 Peter F. Drucker, *Managing the Nonprofit Organization: Principles and Practices*, COLLINS BUSINESS (2005).

169 *Id.*, Jansen & Kilpatrick

170 *Id.*

171 Mikael, *Growing Up*, NODE & JAVASCRIPT: MEDIUM https://medium.com/node-js-javascript/growing-up-27d6cc8b7c53#.jz9yqr74e (May 7, 2015).

172 *Id.*

173 *Id.*

174 Candance Widmer & Susan Houchin, *Governance of National Federated Organizations*, NATIONAL CENTER FOR NONPROFIT BOARDS, (1999).

175 *Id.*

176 *Id.*

177 *Id.*

178 *Id.*

179 Bacon, Jono *The Art of Community*, O'REILLY MEDIA, INC., (2009)

180 *Id.*, at "Setting Up a Community Council," *ch. 8.*

181 Paul J. Jansen & Andrea R. Kilpatrick, *The Dynamic Nonprofit Board*, McKINSEY & COMPANY, http://www.mckinsey.com/industries/social-sector/our-insights/the-dynamic-nonprofit-board, (last visited May 20, 2016).

182 *Id.*

183 Steve McCurley & Rick Lynch, *Volunteer Management: Mobilizing all the Resources of the Community*, HERITAGE ARTS PUBLISHING (1996), at 12.

Case Study #1: Apache Software Foundation

The Apache Software Foundation ("Apache") was founded in 1999[1] to provide an organizational framework and support services for a variety of open source software projects.[2] Apache's most well-known project is the HTTP Server Project, which provides the world's most widely used server software.[3] The vast majority of Apache's revenue comes from donations.[4] Apache's donations web page notes that "donations to [Apache] should be tax-deductible."[5]

Apache applied for 501(c)(3) status in or around mid-2000.[6] After responding to an IRS request for additional information,[7] Apache was granted 501(c)(3) status in February 2001.[8]

In Apache's application for 501(c)(3) status, the organization described its activities in general terms and provided some arguments for linking its work to one of the exempt purposes listed in 26 U.S.C. 501(c)(3). In the narrative description of the organization's activities, Apache emphasized its goals of "supplying hardware, communications, and business infrastructure" for open source software projects; "creat[ing] an independent legal entity to which individuals can donate resources"; and "provid[ing] a means for individual volunteers to be sheltered from legal suits."[9] According to the application, Apache's activities constituted a scientific purpose because "the organization is using its research for the public benefit through the open source software it develops and sponsors."[10]

Apache did not argue that the foundation itself directly produces any public benefits. Instead, Apache portrayed itself as an umbrella organization for open source software projects. In stating its exempt purpose, however, Apache asserted that its exempt purpose could rest on the basis of the research conducted by projects supported by Apache. Because the public might benefit from the projects' software, Apache implies, its research is "for the public benefit." Apache's response to the IRS's request for additional information did not provide any additional arguments to support the assertion that Apache had an exempt purpose, other than by offering greater detail regarding its individual projects.[11] The IRS granted 501(c)(3) status to Apache.

The IRS considers research to be "in the public interest" if the organization in question meets any one of a number of requirements. One way of satisfying these requirements is for the organization to make the results of its research "available to the public on a nondiscriminatory basis."[12] In its application, Apache explained that its software was copyrighted in a way that granted "the general public the maximum flexibility and access to the software and to allow its reuse to as large an extent as possible."[13] Compared with open source projects that have applied more recently, this "public interest" argument is more loosely constructed.

Apache's application process illustrates a trend at the IRS toward more restrictive standards for granting 501(c)(3) status. Apache received 501(c)(3) status less than a year after submitting its original application. The organization managed to survive multiple rounds of review without providing the level of detail or all the kinds of information required by current IRS guidance. The differences in the Apache application process and the specificity required for 501(c)3 status points to a shift in policy and a new understanding of open source organizations at the IRS that has resulted in higher thresholds for success.

---

1 *See* The Apache Software Foundation, Form 1023: Application for Recognition of Exemption Under Section 501(c)(3) of the Internal Revenue Code (2000), *available at* http://www.apache.org/foundation/records/ASF-1023.pdf; I.R.S. Final Determination Ltr. (Feb. 21, 2001), *available at* http://www.apache.org/foundation/records/ASF-501c3.pdf.

2 The Apache Software Foundation, *supra* note 1.

3 *November 2014 Web Server Survey*, Netcraft, http://news.netcraft.com/archives/category/web-server-survey/ (last visited Dec. 5, 2014).

4 *See* The Apache Software Foundation, Form 990: Return of Organization Exempt from Income Tax 1 (Dec. 15, 2013), *available at* http://www.apache.org/foundation/records/990-2012.pdf.

5 *Donating to the Apache Software Foundation*, The Apache Software Foundation, http://www.apache.org/foundation/contributing.html (last visited Dec. 5, 2014).

6 *See* The Apache Software Foundation, *supra* note 1.

7 *See* The Apache Software Foundation, Amendments to Form 1023, available at http://www.apache.org/foundation/records/ASF-1023-Amendments.pdf.

8 *See* I.R.S. Final Determination Ltr., *supra* note 1.

9 The Apache Software Foundation, Form 1023, *supra* note 7.

10 *Id.*

11 *See* The Apache Software Foundation, *supra* note 1.

12 *Id.*

13 The Apache Software Foundation, *supra* note 1, at 8.

Case Study #2: X.Org[14]

Launched in 1997, the X.Org project develops "an open source implementation of the X Window System,"[15] which provides the foundation for graphical user interfaces.[16] In addition to its development activities, the X.Org Foundation ("X.Org") also hosts conferences and internship programs for students.[17] Through its "Endless Vacation of Code" program, students submit project proposals, and those selected get to work full-time over a three- to four-month periods with a stipend of $5,000 to $6,000 and support from a lead member of the organization's technical community.[18] X.Org also offers the same opportunity without funding for non-students. Prior to receiving nonprofit status, the organization had received "significant funding" from Intel, Google, and Oracle.[19] The organization now solicits donations from the general public.[20] X.Org does not have any paid employees.[21]

X.Org submitted its application for 501(c)(3) status in April 2011.[22] The IRS granted X.Org 501(c)(3) status in May 2012.

In its application, X.Org argued that it was organized exclusively for scientific, educational, and charitable purposes but placed greater emphasis on the latter two purposes. The application did not advance a particularly notable case that X.Org had a scientific purpose. The application stated that the X Window System project "focuses on stability and open standards," which could have led to a discussion of how X.Org's research benefits the public.[23] However, X.Org did not frame most of its activities as research in the text, using the word "research" only twice.[24] Nor did the application make a strong case for satisfying one of the scientific purpose requirements. For example, X.Org did not directly address the IRS standard of research "available to the public on a nondiscriminatory basis," focusing instead on its open source licenses.[25]

X.Org made a much stronger argument for having an educational purpose. 557 defines "educational" for the purposes of obtaining 501(c)(3) status as meaning either "[t]he instruction or training of individuals for the purpose of improving or developing their capabilities" or "[t]he instruction of the public on subjects useful to individuals and beneficial to the community."[26] The list of organizations that may qualify as educational includes "[a]n organization whose activities consist of conducting public discussion groups, forums, panels, lectures, or other similar programs."[27]

In its application, X.Org presented its activities in order to match these requirements. For example, X.Org stated that it was founded in order to "ensure that the public has access to educational and scientific literature that furthers their use of the X Window System,"[28] thus demonstrating how X.Org's work creates valuable opportunities for the public to further their understanding of the system. The application also stated that X.Org publishes "human-readable source code, documentation, and materials,"[29] which X.Org argued helps users "improve their ability to design and implement software."[30]

---

14 Note that the IRS revoked X.Org's 501(c)(3) status in 2013 because the organization failed to file its tax returns. *See* Michael Larabel, *X.Org Foundation Loses Its 501(c)(3) Status*, Phoronix (Aug. 23, 2013, 1:22 PM) http://www.phoronix.com/scan.php?page=news_item&px=MTQ0MzU.

15 X.Org Foundation, http://www.x.org/wiki/ (last visited Dec. 5, 2014).

16 *The X Window System™*, X.Org Foundation, http://www.opengroup.org/desktop/x/ (last visited Dec. 5, 2014).

17 *501(c)(3) Status Determination*, X.Org Foundation, http://www.x.org/wiki/Other/Press/501c3StatusDetermination/ (last visited Dec. 5, 2014).

18 *The X.Org Endless Vacation of Code (EVoC)*, X.Org Foundation, http://www.x.org/wiki/XorgEVoC/ (last visited Dec. 5, 2014).

19 X.Org Foundation, *supra* note 203.

20 *Id.*

21 X.Org Foundation, *supra* note 201.

22 Justin C. Colannino, Re: Form 1023 Application for Recognition of Exemption of X.ORG FOUNDATION (Apr. 29, 2011), *available at* http://www.x.org/foundation/irs-form-1023/1-Cover.pdf.

23 X.Org Foundation, Addendum to Form 1023 2 (Apr. 22, 2011), *available at* http://www.x.org/foundation/irs-form-1023/4-ExhibitC.pdf.

24 *Id.*

25 *Id.*

26 *See* Internal Revenue Service, Be on the Look Out List (Aug. 2010); Be on the Look Out List (Nov. 2010); Be on the Look Out List (Feb. 2011); Be on the Look Out List (Feb. 2012); Be on the Look Out List (Jul. 2012) *available at* http://democrats.waysandmeans.house.gov/press-release/new-irs-information-shows-"progressives"-included-bolo-screening-list.

27 *Id.* at 26.

28 X.Org Foundation, *supra* note 15, at 2.

29 *Id.*

30 *Id.*

X.Org additionally asserted that the release and use of source code fit the second definition of "educational" by calling software development "an activity beneficial to the community."[31] In order to buttress this argument, X.Org cited case law in which the Second Circuit observed that a programmer reading code "might use this information to improve personal programming skills."[32] X.Org also highlighted the specific educational activities it sponsors, including the Endless Vacation of Code, conferences, and "active mailing list and wiki" as examples.[33] This description allowed X.Org to fit the model of an educational organization "whose activities consist of conducting public discussion groups, forums, panels, lectures, or other similar programs."

X.Org also established that it has a charitable purpose as a second justification for 501(c)(3) status. According to 557, a charitable organization must show that it is intended to benefit the public interest[34] by advancing education, assisting the poor, or erecting public works.[35] A charitable organization may show that it advances education by offering scholarships.[36]

To that end, X.Org discussed the Endless Vacation of Code program in detail. The application emphasized that students enrolled in the Endless Vacation of Code receive "a monetary reward" upon completion of program goals.[37] X.Org also argued that it assisted the poor by providing its software to the public for free. This benefit is amplified because X.Org code is built into other popular software, driving down development costs, which gives everyone, including the poor, greater access to good software.[38]

Finally, X.Org argued that its software development constitutes the erection of a public work. X.Org defined public works as facilities or processes that are "beneficial to the general public" by generalizing from examples like public swimming pools, playgrounds, or public transportation. Whereas traditional examples of public works are created to benefit a specific community,[39] X.Org's software allows anyone who wants to download its software or incorporates its code to benefit, removing the hampers of "occupancy limitations" or "convenience of physical location."[40] This argument attempts to expand "public works" to accommodate digital products and services offered to the public and unrestricted by license requirements.

X.Org's application provides multiple arguments for federal tax exemption that differed from Apache's approach (addressed in the accompanying Case Study #1). In Apache's application, the organization described its software development activities, briefly asserted that its activities constituted scientific research that benefited the public, and claimed that the organization was founded for an exempt scientific purpose. In contrast, X.Org emphasized its non-development activities, including an internship and scholarship program, and described how open-source software development can fit the concept of public works.

The IRS had already begun to subject open-source software organizations to closer review when X.Org submitted its application.[41] Therefore, X.Org's efforts to cast itself as an educational and charitable organization may have been in response to these heightened standards. Regardless of X.Org's motivations for advancing these arguments, their application demonstrates that open source organizations often have means other than "scientific purposes" to advocate for tax-exempt status.

---

31 *Id.*
32 *Id.* (citing *Universal City Studios, Inc. v. Corley* 273 F.3d 429, 448 (2d Cir. 2001)).
33 *Id.* at 2–3.
34 Internal Revenue Service, *supra* note 26, at 28.
35 *Id.*
36 *Id.*
37 X.Org Foundation, *supra* note 15, at 3.
38 *Id.*
39 *Id.*
40 *Id.*
41 McMillan, *supra* note 9 (discussing OSET's submission of its application for 501(c)(3) status in 2007).

Case Study #3: Yorba Foundation

The Yorba Foundation ("Yorba") is a nonprofit organization that develops Linux desktop software.[42] Its projects include Shotwell, a photo manager;[43] Geary, an IMAP email client;[44] and California, a calendar application.[45] Yorba's projects are hosted by GNOME, but Yorba and the GNOME Foundation are legally and financially independent of one another.[46] Yorba applied for 501(c)(3) status in 2009 and received a final determination letter rejecting the application in 2014.[47]

In its application, Yorba advanced many of the same arguments used by X.Org (described in the accompanying Case Study #2). Yorba claimed that it was organized for charitable and scientific purposes.[48] Like X.Org, Yorba attempted to show that it advanced education, assisted the poor, and erected a public work. Yorba claimed to advance education by making available to the public its source code as well as a wiki and user guide.[49]

Yorba's assertion that "[f]ree and open source software fundamentally has an educational component"[50] was reminiscent of arguments made by X.Org,[51] but unlike X.Org, Yorba did not offer any conferences, internships, or scholarship programs. In explaining how its software assists the poor, Yorba noted that its products provide free alternatives "to software that can sell for as much as $1,000 a license."[52] In contrast, X.Org suggested that its software helped to drive down development costs, without a specific price comparison.[53] Yorba similarly framed free open-source software as a public work because the software, "through free and open source licensing, [is] dedicated to the public."[54] This argument differs slightly from X.Org's argument, which focused on the public actually deriving benefits from X.Org's software. Yorba does not advance a detailed argument in support of its scientific purpose claim, stating simply that "free and open source software project creates a public domain of technical knowledge that anyone can learn and use."[55]

Yorba application for 501(c)(3) status was rejected, and the IRS justified its decision by defining key terms and concepts narrowly, possibly a result of a shift in policy toward open-source organizations. For example, while Yorba and X.Org both claimed to serve the public as a charitable class, in the Yorba determination letter the IRS decided that the public does not qualify because not "all members of public [sic] share any charitable characteristics."[56] The IRS also refused to accept that source code has educational value. The IRS explained that educational activities must involve "instruction" or "training," while Yorba provided its source code "without any additional activity."[57] The IRS contended that the purpose of providing source code to users is to allow them to modify the code and that any learning in that process is "incidental,"[58] undercutting the educational value of source code articulated by both organizations.

The IRS also rejected the proposition that no-cost, non-proprietary software constitutes a public work. The IRS referred to historical sources to define "public works" as including only physical facilities "ordinarily provided at public expense."[59] Based on these sources, the IRS decided that "intangibles" like software do not constitute

---

42 Kendra Albert, *Open Source Madness*, Electronic Frontier Foundation (Jul. 16, 2014), https://www.eff.org/deeplinks/2014/07/open-source-madness.

43 *Shotwell*, GNOME Wiki, https://wiki.gnome.org/Apps/Shotwell (last visited Dec. 5, 2014).

44 *Geary*, GNOME Wiki, https://wiki.gnome.org/Apps/Geary (last visited Dec. 5, 2014).

45 *California*, GNOME Wiki, https://wiki.gnome.org/Apps/California (last visited Dec. 5, 2014).

46 Jim Nelson, *Yorba and GNOME*, Yorba Blog Archives (Dec. 23, 2013), http://blogs.gnome.org/jnelson/2013/12/23/yorba-and-gnome/.

47 Nelson, supra note 2.

48 I.R.S. Final Determination Ltr. 1 (May 22, 2014), *available at* http://yorba.org/docs/IRS-determination-letter-final.pdf.

49 *Id.* at 2.

50 *Id.* at 2.

51 X.Org Foundation, *supra* note 15, at 2.

52 I.R.S. Final Determination Ltr., *supra* note 48, at 2.

53 X.Org Foundation, *supra* note 15, at 3.

54 I.R.S. Final Determination Ltr., *supra* note 48, at 8.

55 *Id.* at 3.

56 *Id.*

57 *Id.* at 9–10.

58 *Id.* at 10.

59 I.R.S. Final Determination Ltr., *supra* note 234, at 8 (citing Comment (k) Restatement 3d Trusts; The Statute of Charitable Uses, 43 Eliz. I, c.4 (1601) (describing as charitable the construction of "bridges, ports, havens, causeways . . . and highways"); *Black's Law Dictionary*, 7th ed. (defining "public works" as "[s]tructures, such as roads or dams built by the government for public use and paid for by public funds")).

public works.[60] Even if intangibles could qualify as public works, the IRS argued they would not qualify because open-source software is not "ordinarily provided at public expense" by governmental agencies.[61] This narrow formulation of "public works" differs significantly from the definitions put forth by Yorba and X.Org that hinge on whether the resource is intended to benefit or actually benefits the public.

Finally, the IRS rejected the argument that open-source software development constitutes scientific research, for which neither organization advanced strong arguments. The IRS classified Yorba's software development as "routine product development," rather than "testing to validate scientific hypotheses."[62] Based on this classification, the IRS found Yorba's activities to be more commercial than scientific. By narrowing terms and reformulating concepts, the IRS raised the threshold an open-source software organization would need to meet in order to obtain 501(c)(3) status.

Yorba's determination letter need not doom nonprofit open-source software development. IRS determination letters have no precedential value.[63] As a result, the conclusions reached in the Yorba determination letter are not binding on subsequent determination letters, though they may provide insight into how the IRS thinks about certain issues at a given point in time. The IRS may reach different or even opposite conclusions in the future. Second, the differences between Yorba's and X.Org's activities may point to the sorts of activities the IRS will look for in deciding whether to grant 501(c)(3) status to open-source software organizations. X.Org offered conferences and a scholarship/internship program to students in order to train and instruct software developers in their use of X.Org's software. Yorba did not offer any comparable educational activities. Learning from the experience of Yorba, an open-source software organization considering 501(c)(3) status may want to consider whether and how they are furthering software development education and emphasize such actions in their application

60 *Id.*
61 *Id.* at 9.
62 *Id.* at 10.
63 26 U.S.C. 6110(k)(3) (2012).

CASE STUDY #4: BRAVE NEW SOFTWARE

Brave New Software ("BNS"), an open-source software organization "dedicated to keeping the Internet open and decentralized" through the creation of tools to combat censorship and other limitations to internet access, submitted a 501(c)3 status application in August of 2010.[64] A letter from the State Department advocating for an expedited process accompanied their application. As is the case with many OSS organizations, the IRS took nearly two years to make an initial determination on BNS's case, issuing a rejection in May 2012. The IRS responded to each of the 501(c)3 purposes BNS argued it qualified for: charitable (and within that, "lessening the burdens of the US government," promoting human and civil rights, "promoting relief of the poor and distressed or underprivileged," and "erection or maintenance of public works"), educational, and scientific.[65]

The IRS contended that several pronouncements by the State Department about their desire to fight internet censorship did not suffice to prove that the work of BNS was related to a burden of the US government. The IRS felt that a State Department grant to BNS also did not indicate that the department needed BNS's help to combat censorship, but more closely resembled a "payment for a service" than burden sharing.[66] Next, the IRS argued that First Amendment and UN international protections of free speech and freedom of expression could not be expanded to include a human right to internet access, and therefore, BNS could not qualify as promoting human and civil rights through their work.

Regarding the claim of "promoting relief of the poor," the IRS determined that because BNS did not limit who could use their censorship circumvention software, their actions did not sufficiently impact a distressed or marginalized group to meet this standard.[67] Confusingly, their rejection of BNS's claim as a creator of a public work stated that BNS's software to combat censorship was too targeted to qualify as a public work (since only those dealing with censorship could benefit from it), and, moreover, that software did not meet the traditional definition of a public work (similar to the determination in Yorba's application).[68]

Turning to the educational purpose, the IRS found that BNS did not educate the public through its publication of code or materials to help understand how to use its software because those were activities that a commercial group might perform to promote its products.[69]

Additionally, the IRS determined that BNS did not promote education through the creation of its software because the censorship circumvention software could be used to access anything online, including potentially illegal or dangerous materials that the IRS believed countries had the right to restrict. Finally, the IRS concluded that BNS did not meet the scientific purpose definition because its activities most closely resembled commercial product research development, not scientific research, and BNS is not a publisher or advocate of scientific scholarly literature.[70]

BNS and its attorneys responded to the denial in June of 2012 and offered arguments to counter most of the IRS' claims, starting with the IRS determination that the US government did not consider itself responsible for promoting internet freedom. BNS pointed to the announcement of the Internet Freedom Program by the State Department, where Secretary Clinton used the word burden to describe the US role in combatting online censorship, as well as other instances when the government talked about the need to support censorship circumvention technologies.[71] Further, BNS argued that the State Department specifically designed their Internet Freedom Program to reflect the fact that the government did not have the means to fulfill their burden in combatting internet censorship.

BNS corrected the IRS's reading of their program's human rights focus, pointing out that "freedom of expression

64 "About Us," Brave New Software (accessed March 15, 2016) http://www.bravenewsoftware.org/; "IRS Ruling Documentation," Open Source Initiative (accessed March 15, 2016) https://wiki.opensource.org/bin/Main/Projects/entities-wg/IRS+Ruling+Documentation/

65 Lerner, Lois G., "Letter of Denial," Internal Revenue Service (May 16, 2012), https://wiki.opensource.org/bin/Main/Projects/entities-wg/IRS+Ruling+Documentation/#HBraveNewSoftwareProject2CInc.

66 Id., at 13.

67 Id.

68 Id.

69 Id.

70 Id.

71 Williamson, Aaron & Adam Fisk, Brave New Software Project Letter of Protest, June 28, 2012

online depends on uncensored access," and that their goal was not to tout internet access as a human right.[72] In response to the IRS's assertion that any group that aids the distressed or underprivileged must make sure their resources were primarily available to the groups in question, BNS used unrestricted internet access at libraries, a traditional class of non-profit, as a counter-example. BNS referenced several well-known occasions when library users or libraries themselves have promoted or engaged in illegal activities, such as endorsing WikiLeaks or allowing the Unabomber to plan his attacks on a public computer.[73] According to BNS, this type of policy is permitted, even if the results are illegal activity, because neither US nor international law uses prior restraint as a policy with regards to online speech, given that such policies often preemptively limit free speech. References to the United States State Department used by the IRS to indicate support for prior restraint were actually discussing the right of states to punish illegal activity after the fact.[74]

Nonetheless, BNS imagined these improper uses would be incidental when measured against uses as intended. BNS again returned to the example of libraries when refuting the IRS's rejection of its activities as educational, reminding the agency that if educational organizations could only provide access to "educational" materials, most libraries would be ineligible.[75] BNS also highlighted the many programs they offered to educate the public on combatting censorship. Finally, the organization indicated the IRS's determination that its work was too broad to qualify as promoting relief of underprivileged groups, but too narrow to qualify as a public work seemed problematic.

The IRS did not respond to BNS's letter of protest until nearly two years later, in June of 2014 when it reversed its decision.[76] Unfortunately, the notice of approval does not contain any information about what occurred in the intervening time or what elements of BNS's letter proved persuasive.

As the IRS notes, each individual IRS application is considered on its merits, and previous approvals or denials are not precedential. In addition, the fact that BNS had substantial support from the US Department of State makes this case unique in certain aspects. Nonetheless, there are some lessons we can draw from this exchange that can aid fledgling OSS organizations considering non-profit status.

As X.Org and Yorba both confirm, organizations should not try to contend that they fulfill a scientific purpose unless their software development contributes directly to scientific research as defined by the IRS – "the organization must (1) engage in scientific research; (2) the scientific research must not include activities that are incident to commercial or industrial operations; and, (3) the scientific research must be undertaken in the public's interest" – or helps disseminate scientific research.[77] While OSS organizations may feel tempted to check as many boxes as possible in a 501(c)3 application, it may be more effective to concentrate on other exempt categories that have a direct tie to the organization's mission. The use of scientific tools does not make an action "scientific" in the eyes of the IRS, and they are unlikely to see anything outside the traditional confines of scientific research and debate as appropriate for such a designation. This line of argument was the only one that BNS abandoned in their response to the IRS' initial decision.

Open source software initiatives can also follow BNS's example in highlighting the parallels between their purpose and services and existing classes of non-profit organizations. In its reply to the IRS, BNS compared the example of libraries providing internet access and non-educational content to visitors to their own program to demonstrate that similar First Amendment reasoning against prior restraint apply in both.[78] This parallel helped illustrate that the IRS's claim that an educational organization should only provide access to educational materials was inconsistent with its view of a large class of charitable organizations.

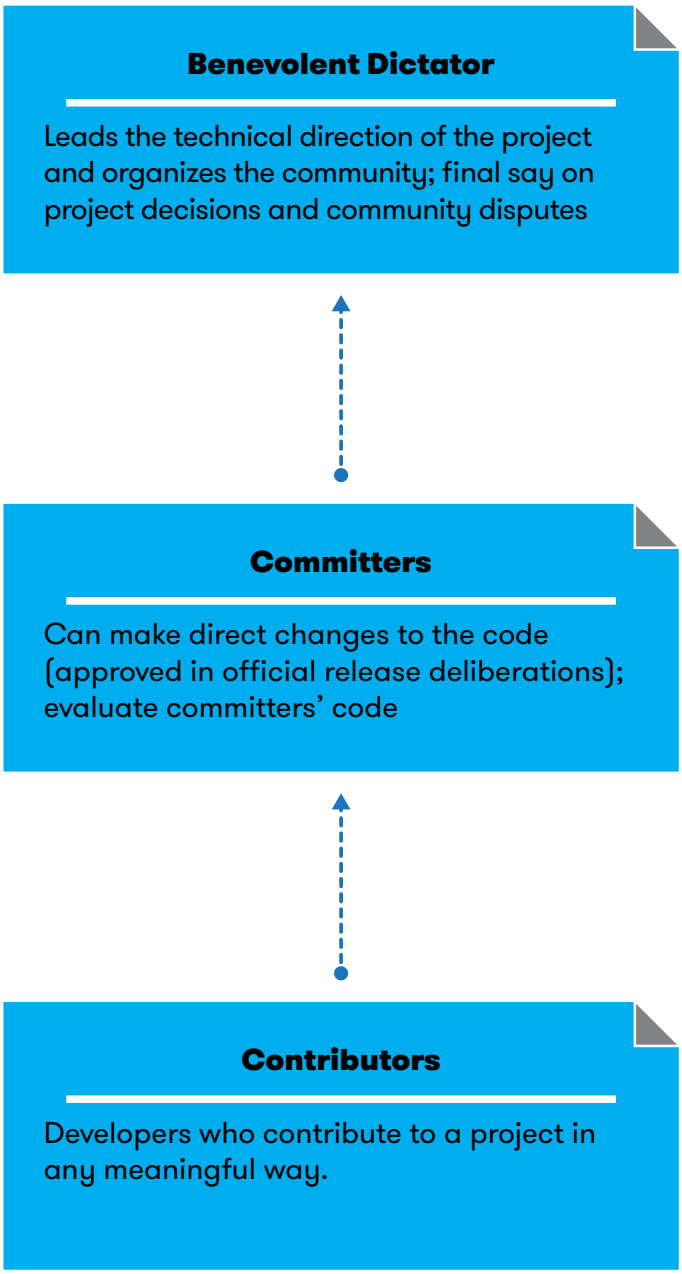
While BNS made this point in fighting back against an initial IRS decision, other organizations could make parallels preemptively to groups that traditionally get IRS approval. Doing so may make the functions of an OSS's services more comprehensible for the IRS staff. Providing analog examples can help move the conversation from the realm of technological capabilities into a discussion of the aims of the organization.

Finally, while few OSS organizations (or non-profits in general) enter the 501(c)3 process with endorsements

72 Id., at 4.
73 Id.
74 Id.
75 Id.
76 Id., Open Source Initiative.
77 Id., Lerner, at 18.
78 Id., Williamson & Fisk.

and a mandate from the federal government, other groups could emulate the tactics BNS used to convey the connection between their mission and an evidenced need in the world. Organizations that are able to cite a specific problem in human rights work, education, or other exempt class and demonstrate how the use of open-source technology can directly remedy that problem stand a better chance of approval than those who focus their discussion on the merits of OSS alone. OSS organizations without a clear connection to real-world problems or human rights implications may find getting 501(c)3 status challenging, given the IRS's skepticism and limited understanding of software development and its potential applications.

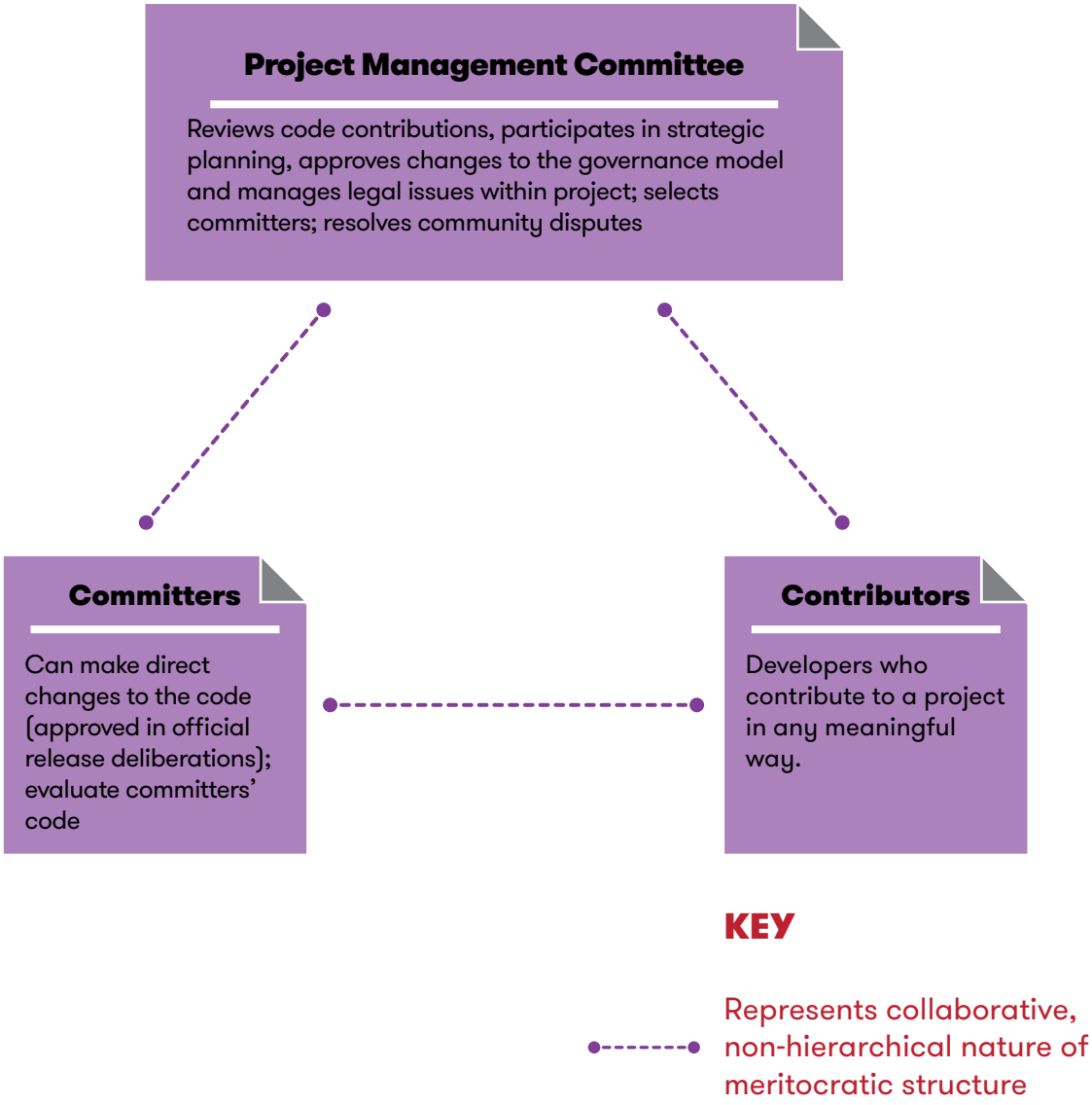
MODEL A:
BENEVOLENT DICTATORSHIP



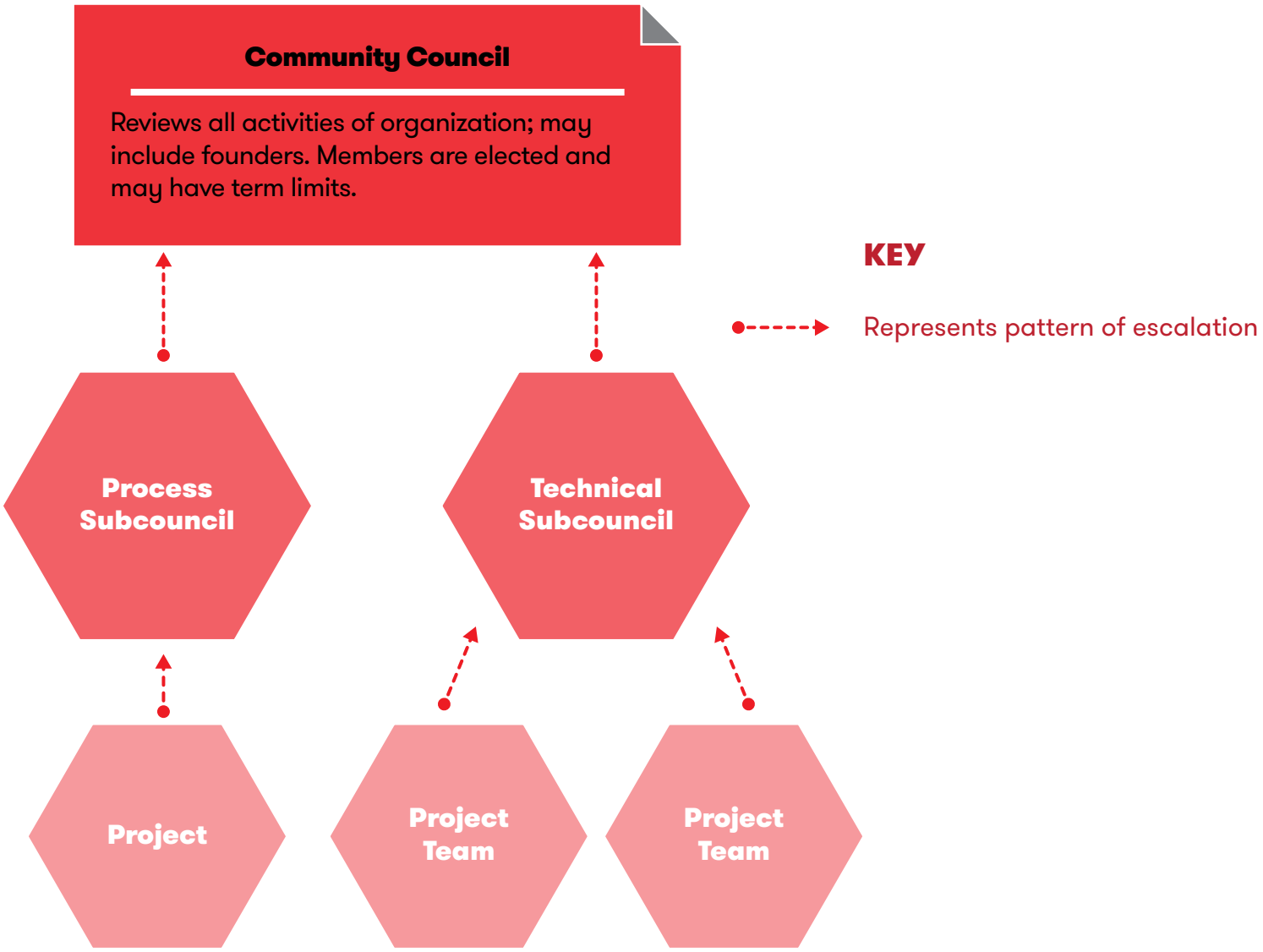
KEY

Represents pattern of escalation

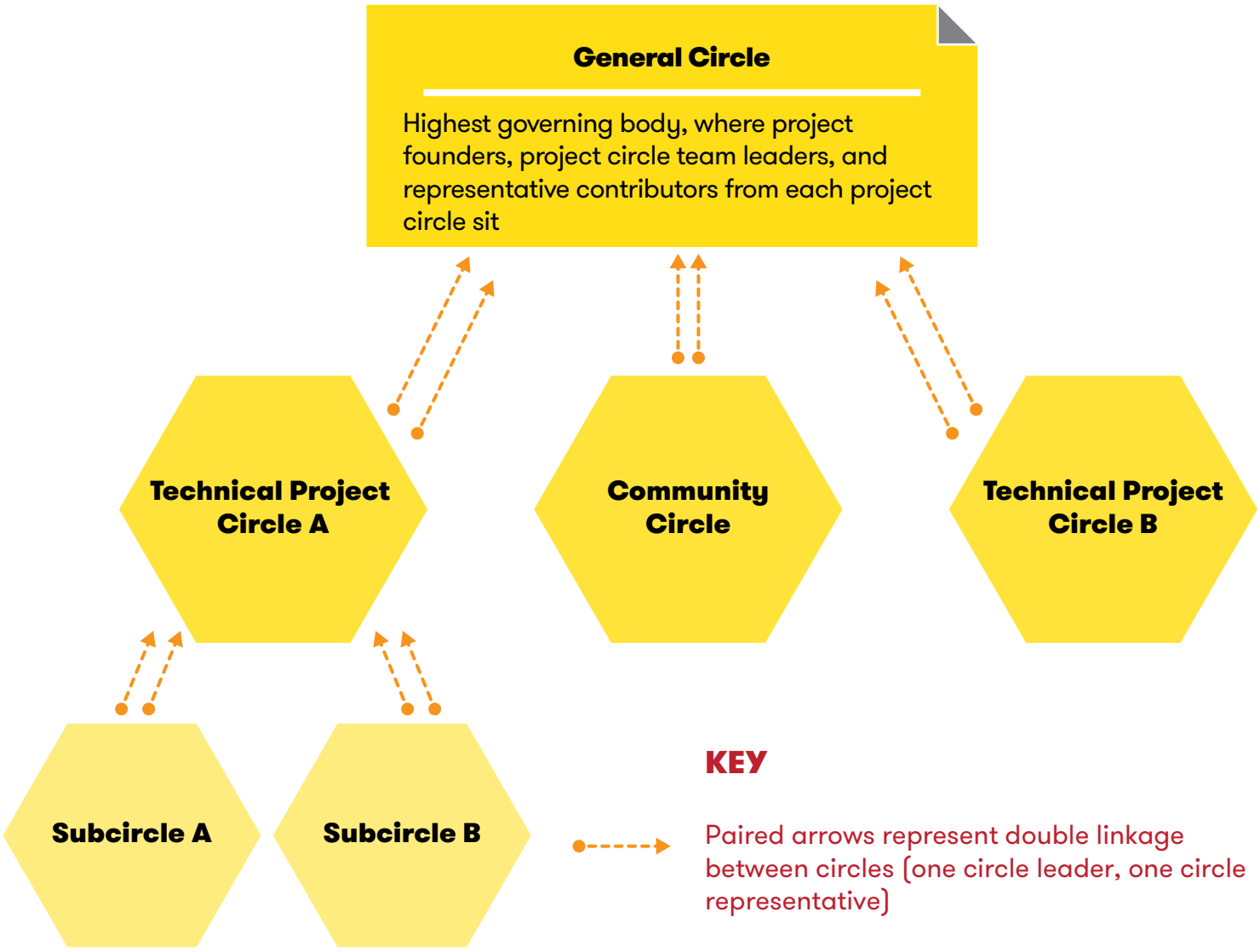
MODEL B: MERITOCRACY



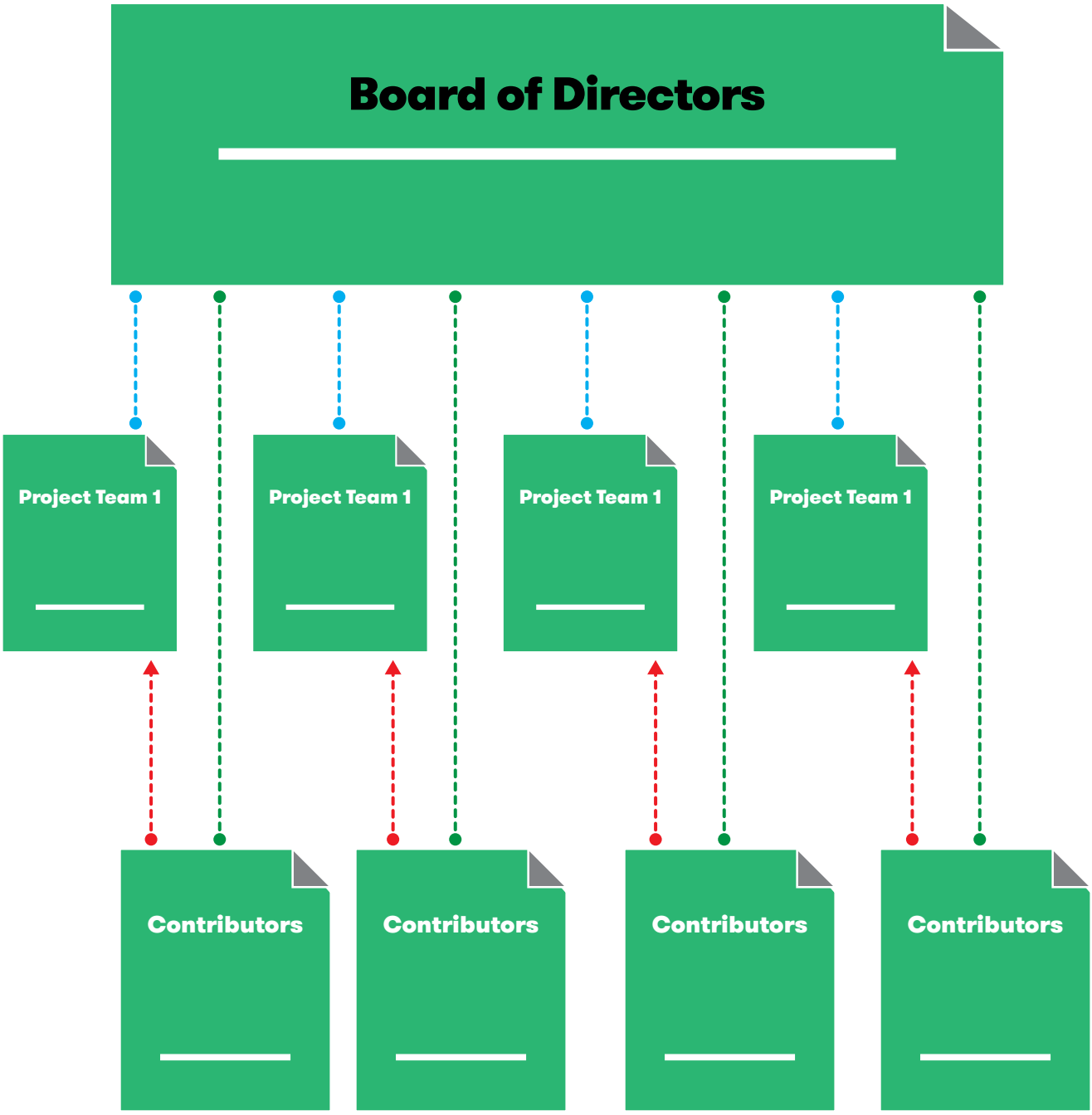
MODEL C: DELEGATED GOVERNANCE



MODEL D: DYNAMIC GOVERNANCE



MODEL E:
FEDERATED NONPROFIT



KEY

Represents escalation from chapter members (contributors) to chapter leadership

Represents collaboration between chapters and executive board of directors

Represents voting and referendum power of contributors at the executive level